UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

Design Patterns for Robot Swarms

by

Lenka Pitonakova

Thesis for the degree of Doctor of Philosophy

February 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
Electronics and Computer Science

Doctor of Philosophy

DESIGN PATTERNS FOR ROBOT SWARMS

by Lenka Pitonakova

Demand for autonomous multi-robot systems, where robots can cooperate with each other without human intervention, is set to grow rapidly in the next decade. Today, technologies such as self-driving cars and fleets of robotic assistants in hospitals and warehouses are being developed and used. In the future, robot swarms could be deployed in retrieval, reconnaissance and construction missions.

Distributed collective systems have desirable properties, such as low cost of individual robots, robustness, fault tolerance and scalability. One of the main challenges in swarm robotics is that 'bottom-up' approach to behaviour design is required. While the swarm performance is specified on the collective level of the swarm, robot designers need to program control algorithms of individual robots, while taking into account complex robot-robot interactions that allow emergence of collective intelligence. In order to be able to develop such systems, we need a methodology that aligns bottom-up design decisions with top-down design specifications.

In this thesis, a novel approach to understanding and designing robot swarms that perform foraging and task allocation is proposed. Based on thousands of different simulation experiments, the Information-Cost-Reward framework is formulated, that relates the way in which a swarm obtains and uses information, to its ability to use that information in order to obtain reward efficiently. Secondly, based on the information-based understanding of swarm performance, design patterns for robot swarms are formalised. The design patterns are modular aspects of robot behaviour that define when and how information should be obtained, exchanged or updated by robots, given particular swarm mission characteristics. Multiple design patterns can be unambiguously combined together in order to create a suitable robot control strategy.

The design patterns specify robot behaviour in a newly developed Behaviour-Data Relations Modeling Language, where relationships between robot behaviour and data stored in and outside of robots are explicitly defined. This allows the design patterns to define behaviour of robots that cooperate and share information.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Lenka Pitonakova , declare that the thesis entitled *Design Patterns for Robot Swarms* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: (Pitonakova et al., 2014), (Pitonakova et al., 2016a), (Pitonakova et al., 2016b)

Signed:.....

Date:.....

Acknowledgements

I would like to thank my partner for much needed encouragement and support.

A big thanks to my supervisors, Seth Bullock and Richard Crowder, for their invaluable help, patience and for teaching me the art of critical self-reflection.

This work was supported by an EPSRC Doctoral Training Centre grant (EP/G03690X/1). I am grateful to the Complex Systems Simulation DTC at the University of Southampton for giving me a chance to pursue my own research interests and for financially supporting me throughout.

Nomenclature

N_W	The number of worksites
N_A	The number of active worksites
N_D	The number of depleted worksites
D	Worksite distance from the base
N_R	The number of robots
ΔI	The information gain of a swarm
i	The information gain rate of a swarm
C_U	The uncertainty cost of a swarm
C_M	The misplacement cost of a swarm
m	The misplacement cost coefficient of a swarm
C_O	The opportunity cost of a swarm

Chapter 1

Introduction

Demand for autonomous physical inter-connected devices is set to grow rapidly in the next decade. The increasing popularity of the "Internet of Things" and of self-driving cars are good examples. In robotics, a lot of effort is being put into autonomous agricultural robots (e.g., Cartade et al., 2012), automated warehouses (e.g., Stiefelhagen et al., 2004; Vivaldini et al., 2010; D'Andrea, 2012) and delivery robots (e.g., Thiel et al., 2009). Finally, the ESA (ESA Mars Missions, 2015; ESA Space Applications Workshop, 2014), NASA (NASA Mars Missions, 2015) and the private space sector (e.g., Google Lunar X Prize, 2015; Astrobotic, 2015) recognise the potential of autonomous robotic teams, that could be sent to the Moon or to Mars in order to explore the environment and terraform it in preparation for a human crew. Despite the significant potential of robot swarms, collective intelligence is currently difficult to understand and design. It is not clear what methodology should be used in order to select an appropriate robot behaviour that, when executed on multiple cooperating robots, delivers a desired collective performance. This thesis considers collective foraging and task allocation with homogeneous robot swarms and formalises nine swarm design patterns, i.e., stand-alone modules of robot behaviour, that can be combined together into appropriate robot control strategies. Swarm mission characteristics are aligned with suitable design patterns by using a novel framework that explains collective performance in terms of flow of information between robots.

Compared to the traditional robotic applications, where a single robot performs all the work, or where multiple robots are controlled by or communicate via a centralised controller (e.g., Stiefelhagen et al., 2004; Vivaldini et al., 2010; Raman, 2014), distributed collective systems have desirable properties such as low cost of individual robots, robustness, fault tolerance and scalability (Brooks, 1989; Mataric et al., 2003; Campbell and Wu, 2011; Fernandez-Marquez et al., 2013). A swarm of robots can exhibit complex collective behaviour, such as distributing itself across worksites according to their importance, or adjusting the number of working robots based on the amount of work available, without the need for a "supervisor" in the loop. This makes robot swarms

ideal candidates for autonomous long-term work that requires minimal human intervention. In the future, robot swarms could be used for mining operations in hard-to-reach areas, for securing and cleaning up nuclear power plants that underwent meltdown, in collective construction and for other tasks that are difficult for people to perform.

Unlike conventional engineering, multi-robot engineering requires a "bottom-up" approach to behavioural design (Trianni et al., 2011; Parunak and Brueckner, 2015). The emergent macro behaviour of the swarm is specified and evaluated, but it is the micro behaviour of individual robots that needs to be programmed. While many swarm robotic experiments have been performed both in simulation and in the real world, a framework that organises and explains the results is still missing. Robot swarms are usually designed and optimized for specific tasks, after rather arbitrary decisions have been made about the nature of their control algorithms. Furthermore, it is often unclear which design decisions can be generalized to other tasks or why swarms designed in a particular way exhibit the behavior that they do. Several swarm robotics experts (e.g., Nagpal, 2004; Parunak and Brueckner, 2004; Serugendo et al., 2006; Winfield, 2009; Gardelli et al., 2007; Reina et al., 2015) have recognized that in order to be able to design robot swarms reliably, efficiently and safely, we need an implementation-generic algorithm design methodology that aligns bottom-up design decisions with top-down design specifications.

In software engineering, a design pattern associates a particular class of known problem with a particular class of effective solution. Object-oriented design patterns (Gamma et al., 1994) transformed the field of software engineering and allowed it to grow rapidly as an industry by providing verified solutions to recurring problems. Similarly, robot design patterns could help to establish a mature methodology for the field of swarm robotics and bring it closer to practical applications. Some swarm design patterns have already been identified in the context of multi-robot self-assembly (Nagpal, 2004) and ant-inspired foraging (Gardelli et al., 2007). Other work focuses on a design methodology for collective decision-making (Reina et al., 2014). However, the lack of abstract description and formalisation of these patterns means that it is not clear what class of problems they could be applied to. Furthermore, because no single established framework for engineering swarm intelligence exists, researchers often do not agree on what the level of description for robot design patterns should be. While some concentrate on detailed implementation of robot behavior (Fernandez-Marquez et al., 2013), others describe macro capabilities of swarms (Gardelli et al., 2007).

It is proposed in this thesis that design patterns should be represented in terms of "modules" that act as templates for different aspects of robot behaviour and identify how that behaviour affects the swarm's ability to obtain, process and utilise information[1]. Because swarm performance is understood at a relatively abstract level, in terms of information flow between robots, the design patterns are hardware- and software-generic and thus

reusable across different swarm missions. Given particular mission requirements, different design patterns can be combined together to form a robot *control strategy*.

Two robot tasks are explored: the *consumption* task and the *collection* task. In the *consumption* task, robots search for worksites in the environment and obtain reward by staying close to the worksites while gradually depleting them. In the *collection* task, robots extract small packets of resource from worksites and deliver them to the base. They thus have to make a number of foraging trips in order to deplete a single worksite. The tasks are explored in a number of *static* and *dynamic* environments. In *static* environments, worksite locations are determined at the beginning of an experimental run. In *dynamic* environments, worksite locations change over time. The tasks and environments investigated here model a number of real-world swarm-robotic missions, such as cleaning of streets, collection of raw materials, picking up and delivering of packages or maintenance of a warehouse or of a shop floor.

There are three novel contributions of this work. Firstly, the *Information-Cost-Reward (ICR) framework* is developed, providing an abstract, information-based account of swarm behavior. It is shown that this framework can be used to explain and, to a certain extent, predict performance of homogeneous swarms that utilise various decentralised swarm control strategies previously explored in the literature (e.g., in Sugawara and Watanabe, 2002; Rybski et al., 2007; Campo and Dorigo, 2007; Lemmens et al., 2008; Gutiérrez et al., 2010). Under the ICR framework, a robot swarm is viewed as a single entity that searches for information about worksites in the environment and utilises the information with a certain efficiency in order to obtain reward. A swarm perceives and acts within a perception-action loop - it explores and exploits the environment, changing it in ways that affect how it obtains and utilises new information.

Secondly, nine swarm design patterns are identified and their impact on macro-level swarm behaviour is explained using the ICR framework. The design patterns describe key aspects of robot behavior and are modular in nature. Each pattern identifies what type of robot mission it is suitable for, how robots that utilise the pattern should behave, how the design pattern interacts with other behaviours of robots and what consequences, in terms of information acquisition and utilisation, the pattern has on the macro swarm behaviour. The design patterns are of three types: *transmitter patterns*, that determine how robots share and store information about worksites, *exchange patterns*, that determine where in the environment the information should be exchanged, and *update patterns*, that determine how the information should be updated.

Thirdly, the *Behaviour-Data Relations Modeling Language* (BDRML) is defined. The language allows design pattern creators to unambiguously represent robot behaviour defined by their design patterns, both visually and textually, without the need to specify

[1]The term "information" is used in this thesis to mean "a piece of data" and it is not related to Shannon information as it is traditionally used in engineering. See Section 4.2.2 for further details.

programming code implementation on a particular robot hardware. Unlike in other modeling languages, the relationships between robot behaviour and data structures are expressed explicitly in BDRML, including relationships between behaviour of one robot and data of another. This allows BDRML to easily represent control algorithms where robots cooperate and share information with each other. Furthermore, the language facilitates specification of explicit rules for combining multiple design patterns into a control strategy.

As is often the case in the swarm robotic literature (e.g., Lemmens et al., 2008; Hecker et al., 2012; Schmickl et al., 2012; Hoff et al., 2013), many of the robot behaviours explored here are nature-inspired. Therefore, various animal foraging strategies found in nature are first described in Chapter 2. The chapter continues by discussing the state of the art in swarm robotics and concludes by describing design patterns in the context of object-oriented software engineering and multi-agent systems. Chapter 3 introduces the simulation environment, basic robot control strategies and analysis methods used for all experiments discussed in this thesis. Robot control algorithms are analysed and their performance compared initially in static environments, where worksites do not change during a simulation run (Chapters 4 and 5), as well as in dynamic environments, where worksite locations change periodically (Chapter 6). Chapter 4 also introduces the ICR framework. Two biologically inspired add-on control strategies, that extend the basic control strategies in order to help swarms deal with dynamic environments, are explored in Chapters 7 and 8. Finally, the general lessons learned from all experiments are summarised, the methodology for design pattern creation and BDRML representation is described, and swarm design patterns are formalised in Chapter 9. The chapter concludes with a discussion on how the design patterns catalogue can be extended based on experiments performed by other authors (e.g., by Mayet et al., 2010; Hoff et al., 2010; Ducatelle et al., 2011; Fujisawa et al., 2014). Chapter 10 considers design patterns as a methodology for swarm robotics design, addressing advantages and disadvantages of this approach, and identifying relevant future research directions.

Chapter 2

Background and related work

Many robot control strategies used throughout this thesis and in the swarm robotics literature are inspired by foraging strategies found in nature. During foraging, robots search for worksites in the environment, extract material from them and deliver it to a base in order to obtain reward. Foraging is used as a paradigm for studying a broad range of collective robot behaviour, such as item pick up and delivery (e.g. Wawerla and Vaughan, 2010), task allocation (e.g., Jevtic et al., 2012), labour division (e.g., Zahadat et al., 2013), dispersion (e.g., Ranjbar-Sahraei et al., 2012) and aggregation (e.g., Schmickl and Hamann, 2010). In the first part of this chapter, an overview of animal foraging strategies is thus given, including solitary foraging, behavioural matching, stigmergy-based recruitment, direct signaling and coordinated group hunting. Optimal foraging theory, that is used by both biologists and swarm engineers to reason about swarm performance, is then explained. A review of simulated and real-world experiments with robot swarms, including swarms without communication between robots, swarms with short-range communication, as well as ant-inspired and bee-inspired swarms, follows. Special attention is paid to robot control algorithms that are able to cope with dynamic environments, where the swarm needs to respond to changes in worksite density or priority, since such experiments closely resemble real-world robot missions, for example, pick up and delivery of items, disaster response, etc. Possible future directions of swarm robotic research are then discussed, and it is argued that a working understanding of swarm intelligence is needed in order to make robot swarms applicable in real-world tasks. Finally, introduction to object-oriented design patterns is given and design patterns currently used in multi-agent software and multi-robot systems are discussed.

2.1 Foraging in nature

Studying foraging strategies found in nature can help us understand both the mechanics and the environmental preconditions of collective intelligence and thus guide design of artificial swarms (see Section 2.3). Animal foraging strategies evolved for particular niches, characterised by food abundance and food availability dynamics, by the energetic returns from foraging and by prey handling requirements. For example, animals prefer solitary foraging when their food, such as insects or fruit, is scattered or appears in small portions that satisfy only a fraction of an individual's nutritional requirements. On the other hand, when prey occurs in patches, for example when nectar is extracted from groups of flowers, or when prey aggregate together, different forms of direct and indirect communication between group-living individuals are utilised. Finally, some higher mammals, capable of producing and processing complex signals, hunt cooperatively and share their prey after successful hunts.

2.1.1 Solitary foraging

Most species, especially carnivores, perform solitary foraging (Gittleman, 1989), during which an individual searches for food alone and does not receive or share information with others (Robinson and Holmes, 1982; Darimont et al., 2003). Rather than providing an exhaustive list, this section briefly introduces cases when animals that are normally social foragers choose to forage alone.

Forest birds solitarily forage when various types of insect prey are scattered in patches that are not sufficiently large to sustain a single individual (Robinson and Holmes, 1982). Large enough prey is searched for during flight, pattern of which is set to provide an efficient trade-off between movement and feeding. Environmental cues like curled leaves are searched for when prey is small and hidden. Thoroughness of the search increases as food becomes less abundant. Frigate birds that feed on infrequently scattered fish also forage solitarily, despite the fact that they live in colonies. The bird population tends to spread across foraging areas in order to maximise an individual's feeding rate (Weimerskirch et al., 2004). Furthermore, a single bird never returns to a location where it previously fed, anticipating an unsuccessful search. Tendencies to forage alone during low food abundance (Holekamp et al., 2012) or when food is abundant but offers insufficient portions (Hayward, 2006) were also found in hyenas that are otherwise social.

Some animals, that normally catch difficult prey in groups, also forage solitarily when prey is easy to acquire. For example, lionesses that feed on vulnerable neonates (Stander and Albon, 1993) and wolves that forage for easy-to-catch salmon (Darimont et al., 2003) prefer to obtain such food alone. Chimpanzees also act alone to catch ants (Möbius et al., 2008) or in some occasions to chase and kill small baboons (Busse, 1978).

2.1.2 Behavioural matching

When performing behavioural matching, an individual follows successful foragers and thus utilises social information in order to take advantage of food that is scarce but occurs in patches (Galef and Wigmore, 1983; Clark and Mangel, 1984; Weimerskirch et al., 2004; Webster and Laland, 2012). Several types of behavioural matching in relation to social information processing and learning have been identified (Noble and Todd, 2002). Social facilitation ("Do not do anything unless others are nearby") and contagious behaviour ("Seeing others do something means that I am going to do it") are the most relevant to foraging.

Typical examples of animals that utilise behavioural matching are birds that move in flocks and fish that live in shoals. Individuals join large groups when searching for food (Lachlan et al., 1998) and tend to head towards places where they see others feed (Lachlan et al., 1998; Kendal, 2004; Ward et al., 2012; Webster and Laland, 2012). It has been observed that fish are more likely to join a group when they are uncertain about their own information about food sites (Webster and Laland, 2012) or when the cost of obtaining food based on non-social information is high (Kendal, 2004). Behavioural matching is also used by sheep that search for profitable grass patches to graze on (Michelena et al., 2010).

While birds, fish and grazers consume their food immediately and continue to search for more food randomly, rats live together and gain information about profitable locations by sniffing other rats returning from foraging trips (Galef and Wigmore, 1983). Similarly, socially-foraging birds like ospreys (Flemming et al., 1992), tits, woodpeckers (Sridhar et al., 2009) or ravens (Flemming et al., 1992) head from their nests towards places where they see other birds coming from with prey. The sites where individuals aggregate and gain information are thus considered "information centres" (Ward and Zahavi, 1973).

It is important to point out that behavioural matching is not a cooperative foraging strategy, since information is not shared actively by successful foragers (Galef and Wigmore, 1983). Nevertheless, this foraging behaviour increases the probability of foraging success (Flemming et al., 1992; Hoogland, 1981) by decreasing the risk of individual-based errors (Ward et al., 2012) and the cost of environmental sampling (Flemming et al., 1992; Webster and Laland, 2012). Behavioural matching also reduces the variance of an individual's feeding rate (Clark and Mangel, 1984). When foraging groups are large, for example as it may be the case for bird flocks, this foraging strategy can lead to *information cascades*, where information is gradually propagated through tens or hundreds of individuals without a reference to the original cues (Bikhchandani et al., 1992). Information cascades can have negative effects, especially outdated information and error propagation (Giraldeau et al., 2002).

2.1.3 Recruitment via stigmergy

Stigmergy refers to indirect communication between individuals, where the environment is used as a medium for storing information. Several species of social insects, most notably ants and termites, create chemical pheromone trails from their nest to food sources, helping their nest mates navigate the environment and find nutrition (Beekman, 2001; Barbani, 2003; Ribeiro et al., 2009). A pheromone trail is reinforced each time an individual travels between nest and food and releases pheromone from its body. Trails that are the strongest have the highest probability of being followed (Beekman, 2001; Arab et al., 2012). Over time, a "map" that identifies the most commonly visited, and thus the most profitable, food patches builds up. Unused trails gradually evaporate, assuring that old information eventually disappears from the environment and is not followed.

This strategy can only be utilised by social insect colonies that are large enough (Beekman, 2001; Barbani, 2003), as a sufficient frequency of interactions between group members and trails is required (Berthouze and Lorenzi, 2008). The individuals are often blind and follow a trail to their death (Ribeiro et al., 2009), but facilitation of stigmergy was also observed in addition to using visual cues, gravity, magnetic field (Ribeiro et al., 2009), odour sensing or following the shape of the traversed environment (Barbani, 2003).

Some insects have an especially complicated repertoire of pheromones. The main foraging pheromone is often very volatile but strong and thus rapidly attracts individuals (Arab et al., 2012). Its relatively quick evaporation rate provides a sufficient negative feedback to recruitment (Grace and Campora, 2005; Ratnieks, 2008) and prevents the insects from travelling to locations where a source has been depleted. Furthermore, both ants and termites use weak but long-lasting pheromone while exploring, probably to create a "memory" of the environment that can be utilised to easily reevaluate previously depleted food patches, or in order to build up a recruitment trail quickly when food is found (Beekman, 2001; Arab et al., 2012). Pharaoh ants also use a "no-entry" pheromone that signals others not to follow an established branch of a trail network, allowing a colony to stop following a non-profitable branch more quickly than the pheromone evaporation rate allows (Robinson et al., 2005).

Apart from food quality, direction to the nest is also encoded in the trails. Ants can perceive angles between branches and head either towards or away from the nest (Robinson et al., 2005; Berthouze and Lorenzi, 2008). On the other hand, termites place special long-lived chemicals into their pheromone trails in order to orientate themselves (Arab et al., 2012; Grace and Campora, 2005).

It is notable that the success of stigmergy is very time-dependent. It was shown that ants utilise the fact that trails to a better food source simply take less time to build

up (Sumpter and Beekman, 2003), suggesting that evolution has optimised the balance between the movement speed of ants and the pheromone evaporation rate. As a consequence of this time-dependency, a colony finds it difficult to create a new, shorter, route to a food source if a longer, established, trail already exists (Ribeiro et al., 2009). Another important factor is the number of individuals following a trail. Ants tend to push each other away if there are too many of them at an intersection, which results in creation of additional non-optimal paths (Dussutour et al., 2004).

2.1.4 Recruitment via direct signalling

The direct signaling foraging strategy is typical for social insects that either live in small colonies or fly and thus cannot utilise stigmergy in order to exchange information about where food is located. Members of a colony usually travel between the nest and food sources and exchange information while they are in or near the nest.

Recruitment by touching antennae has been observed in ant colonies that are small (Beekman, 2001) or forage for food that is scattered by wind as opposed to occurring in patches (Prabhakar et al., 2012). In some cases, direct contact can be used to enforce pheromone trail following (Beekman, 2001). Similarly to stigmergy, a richer or easier-to-obtain food source has a higher frequency of returning foragers, attracting more recruits. However, unlike stigmergy, direct signalling requires both the signaller and the receiver to be present at the same place at the same time, which leads to a highly non-linear relationship between the structure of the nest and the ability of a forager to meet and share its information with its nest mates (Prabhakar et al., 2012).

The most typically studied species that utilises direct recruitment are bees. In contrast with ants and termites, bees depend on flowers, the quality of which can change rapidly (Granovskiy et al., 2012), and have thus evolved to be able to switch quickly to a newly discovered, better food source (De Vries and Biesmeijer, 2002). Recruitment *waggle dances* are performed for nest mates that are interested in foraging in a designated area in the hive, called *the dance floor* (Seeley et al., 1991; Biesmeijer and De Vries, 2001). While the length and strength of a waggle dance are related to the quality of a particular flower patch, the position and orientation of a bee on the dance floor encodes the location of a patch relative to the hive, allowing recruits to travel to specific advertised locations. An individual's decisions about whether to waggle dance for, forage from, or abandon a patch are affected by olfactory and taste information of nectar samples obtained from other foragers through *trophallaxis*, where one bee feeds a small sample of its nectar to another bee (De Marco and Farina, 2003; Farina et al., 2005). For example, when a forager discovers that other bees are processing nectar of a much better quality, it abandons its own source faster (De Marco and Farina, 2001). It has been argued that trophallaxis is an important communication channel when flower quality changes rapidly, as it allows information about patch quality to spread through the whole hive within hours, while

waggle dancing only affects bees that follow dances and is thus a slower communication method (Farina et al., 2005). Bee colonies achieve additional flexibility, that allows them to respond to changes in the environment, through opportunistic scouting, when a recruited forager gets lost due to errors in waggle dance signal propagation (Seeley, 1994). Bees also *inspect*, i.e., occasionally re-evaluate, previously abandoned flower patches, since nectar profitability of flowers can increase under appropriate weather conditions (Granovskiy et al., 2012).

2.1.5 Group hunting

During group hunting, individuals use a range of cooperation methods in order to obtain prey. Engagement in a group hunt means that an individual does not have to rely on food sources of lower profitability (Hoogland, 1981; Skinner et al., 1995; Hayward, 2006; Smith, 2010). The prey is either larger or faster than the hunter itself, as it is for example in the case of lions (Stander and Albon, 1993) or hyenas (Hayward, 2006), and hunting in a group significantly increases the probability of feeding. Alternatively, animals such as dolphins (Benoit-Bird and Au, 2009b), Belukha whales (Bel'kovitch and Sh'ekotov, 1993) and humans (Bliege Bird et al., 2001), cooperatively herd prey into dense clusters, increasing their energetic return from foraging.

The least communication-intensive form of collective hunting is *coordinated* hunting, where no direct signals are passed between group members and each member of the group simply tries to maximise its own chance of obtaining food. Such hunting strategy has been observed, for example, in chimpanzees that prey on monkeys (Busse, 1978; Newton-Fisher, 2007) and hyenas that chase weak members of ungulate herds (Hayward, 2006).

On the other hand, *cooperative* hunters actively synchronise their behaviour in order to increase the chance for success of the group as a whole (Busse, 1978; Dugatkin et al., 1992). Cooperative hunting requires established hunting roles, direct signalling between individuals and rules about how food is shared between group members (Benoit-Bird and Au, 2009a; Smith, 2010). This strategy is used by some chimpanzees (Newton-Fisher, 2007) and lions (Stander and Albon, 1993). Dolphins are able to maximise their feeding efficiency by cooperatively herding small fish into a tight circle (Benoit-Bird and Au, 2009a), utilising short-distance sound signals (Benoit-Bird and Au, 2009b). Similar behaviour was observed in Belukha whales that use echolation pulses while they cooperatively herd fish together and then feed on them (Bel'kovitch and Sh'ekotov, 1993).

Humans mastered group hunting during their evolutionary history, usually relying on centralised cooperative strategies with established leadership roles (Furniss, 1974; Dekker, 2006). Similarly to other animals that forage in groups, we prefer prey that has a high

energetic return rate (Bliege Bird et al., 2001). Our intellectual, linguistic (Smith, 2010) and empathic (Delton and Robertson, 2012) capabilities allow us to decrease the variability of an individual's feeding rate by achieving an effective cooperation during the hunt (Bliege Bird et al., 2001) and by creating "resource pools" that non-successful foragers can temporarily utilise (Smith, 2010).

2.1.6 Summary

As we have seen, there are a large number of foraging strategies in natural systems and it is likely that each one is suited to a particular type of foraging environment. If roboticists are to take inspiration from biological systems, the nature of their fit to particular environmental challenges must be understood.

Solitary foraging is used by most animals as their only foraging strategy (Gittleman, 1989). Moreover, animals that normally forage in groups prefer to find food on their own when their prey is easy to catch or when it is so small that it cannot feed more than one individual.

Behavioural matching is an indirect type of information transfer, where individuals, for example birds (Flemming et al., 1992; Sridhar et al., 2009) or fish (Ward et al., 2012; Webster and Laland, 2012), do not directly communicate with each other, but live in groups and observe each other's behaviour. Successful foragers are followed by other group members to where food can be found, which increases the feeding rate of the uninformed individuals. However, behavioural matching can also have negative consequences, such as propagation of outdated or erroneous information.

Ants (Barbani, 2003; Ribeiro et al., 2009) and termites (Arab et al., 2012) live in colonies and create pheromone trails in order to recruit each other to food. The trails spread out from the nest and provide a "map" of the environment that uninformed individuals can follow. Often, the only source of negative feedback is pheromone evaporation, meaning that new, more optimal paths are difficult to form once a path has already been established.

Other social insects, for example honey bees (Seeley et al., 1991; De Vries and Biesmeijer, 2002), exchange information about food patches in a one-to-one fashion via direct signaling in their nest. The signals used by bees encode both quality and location of food sources, allowing the colony to concentrate on the most profitable flower patches and to also rapidly change its foraging effort as a response to changing environmental conditions.

Group hunting is a foraging strategy mostly utilised by higher mammals, such as hyenas (Hayward, 2006), chimpanzees (Newton-Fisher, 2007), dolphins (Benoit-Bird and Au, 2009b) and whales (Bel'kovitch and Sh'ekotov, 1993). During coordinated hunting,

multiple individuals chase prey together but only try to maximise their own chance of obtaining food. During cooperative hunting, signaling between individuals with established hunting roles is used while chasing or herding prey together, which increases the energetic return from foraging of each group member.

Many robot swarms have been inspired by foraging animals. Solitary robots (see Section 2.3.1) are often used when a simple foraging algorithm is needed in order to implement basic swarm behaviour, that is extended by (e.g., Krieger and Billeter, 2000; Campo and Dorigo, 2007; Labella et al., 2006) or compared with (e.g., Balch and Arkin, 1994; Rybski et al., 2007; Gutiérrez et al., 2010; Lee et al., 2013) other behaviours. Robot control strategies that achieve swarm behaviour similar to behavioural matching (see Section 2.3.2) are used in swarms where robots need to communicate worksite locations to each other any time when they meet (Gutiérrez et al., 2010; Sarker and Dahl, 2011; Miletitch et al., 2013), while bee-inspired robots (see Section 2.3.3) exchange information in the base (Lee and Ahn, 2011; Lemmens et al., 2008; Hecker et al., 2012). Finally, ant-inspired robots (see Section 2.3.4) use chemicals, such as alcohol, or cues dropped in the environment in order to create trails to worksites (e.g., Hrolenok et al., 2010; Mayet et al., 2010; Fujisawa et al., 2014).

2.2 Optimal foraging theory

The optimal foraging theory (OFT) represents a set of analytical models that describe how an optimal forager, i.e., one that maximises its net energy intake rate, should divide its foraging effort between exploration and feeding when food is organised in randomly distributed patches of different types, each type containing a certain number of food "portions". It is assumed that the forager is able to consume food at a certain rate, magnitude of which declines with time spent in a patch.

The net energy intake rate E_n is defined as (Charnov, 1976):

$$E_n = \frac{E_e - t \times E_T}{T_u} \tag{2.1}$$

where E_e is the average energy obtained from a patch, t is the inter-patch travel time, E_T is the energy cost per time unit of traveling between patches and T_u is the average time it takes to consume the whole patch.

A marginal capture rate of a patch is defined as $\delta g/\delta T$, where g is the assimilated energy corrected for the energy cost incurred by searching for the patch and T is the time that a forager spends feeding from the patch. According to the marginal value theorem (Charnov, 1976), a forager should resume exploration when the marginal capture rate of a patch drops to the average capture rate of the environment. Consequently, the

theorem predicts that T should be constant across patches of different types in a given environment, but that it should decrease when the environment becomes more rich. Charnov discusses experiments that showed how the foraging behaviour of birds validates these predictions.

OFT also describes when it is advantageous for an individual to be a part of a flock, where foragers travel together and utilise behavioural matching (see Section 2.1.2). There is a trade-off between taking advantage of social information, which generally increases the individual's intake rate, and sharing information with others, which can lead to interference between group members and thus decrease the energy intake rate. A group member obtains m/n portions from a patch, where n is the group size and m is the total number of portions in the patch. Flocks of size $n > m$ are unstable, as individuals are better-off foraging alone (Clark and Mangel, 1984). According to the model, flocking reduces the length and the variance of the expected time interval between patch discoveries (Ranta et al., 1993) and significantly increases feeding rates (Clark and Mangel, 1986).

Ideal free distribution (IFD) (Fretwell, 1972) describes the optimal distribution of foragers in a multi-patch scenario with patches of varied reward. The IFD for a two-patch scenario is defined as (Fagen, 1987):

$$\log \frac{N_A}{N_B} = \frac{1}{k} \log \frac{F_A}{F_B} \tag{2.2}$$

where N_A and N_B are the number of foragers in patches A and B, $k\epsilon[0, 1]$ is an interference constant and F_A, F_B are resource densities of the patches. IFD is related to the matching law (Herrnstein, 1961), according to which foragers distribute their effort between patches proportionally to patch return rates. Evidence of such behaviour has been found in ant (Sumpter and Beekman, 2003) and bee (Seeley et al., 1991) colonies that were presented with multiple patch choices of varied quality.

In robotics, OFT has been used in order to evaluate how close foraging performance of a swarm is to an optimal performance in a given environment (e.g. Ulam and Balch, 2004; Lerman et al., 2006; Campo and Dorigo, 2007; Liu and Winfield, 2010; Kernbach et al., 2012). However, OFT cannot directly guide behaviour of robots in order to help them achieve a desired performance, since the model relies on global knowledge of worksite distribution and qualities (Kernbach et al., 2012). The use of OFT in robotics is thus limited to parameter optimisation.

2.3 Robot foraging

A wide range of approaches have been taken to implementing foraging robot swarms. In this section, the most common types of homogeneous decentralised robot swarms are

described, including swarms of non-communicating robots, "bucket-brigading" robots, swarms where robots communicate in a peer-to-peer fashion in order to share information about worksites, as well as bee- and ant-inspired swarms. Furthermore, specific approaches to implementing robot swarms capable of responding to environmental changes, such as varying worksite density or utility, are reviewed. The section ends by discussing the future of swarm robotic research and applications.

2.3.1 Swarms without communication

Non-communicating robots have been used in experiments where a simple foraging behaviour was investigated in conjunction with other behaviours, such as task allocation or swarm self-regulation. For example, several studies have explored tasking robots with maintaining a certain amount of "food" items in their base while the items spontaneously disappeared and new ones needed to be foraged for. Using the *response threshold model*, according to which a robot only left the base in order to forage when the number of items in the base dropped below a specific threshold (Krieger and Billeter, 2000; Yang et al., 2009), if the robot was previously successful in foraging (Labella et al., 2006), or if the perceived density of robots and items in the environment was satisfactory (Campo and Dorigo, 2007), robot swarms were able to achieve energy-efficient behaviour by only leaving the base when it was necessary. In a different experiment, robots had to forage for items of two different types in order to satisfy their own "needs" for each item type (Spier and McFarland, 1997; Jones and Mataric, 2003). Without using communication, the robots could do this reliably by using their own internal memory of the environment and feedback loops related to their "needs".

Non-communicating robot swarms were also used in a mission where the swarm needed to distribute itself between multiple worksites proportionally to the worksite utilities (Kernbach et al., 2013). The worksites were represented by light sources, the intensity of which signified the worksite degree of utility. The robots could sense light intensity around them, as well as the presence of other nearby robots. Small swarms, that could fit under a single strongest light source fully aggregated under it, while larger swarms distributed themselves proportionally to the light source intensities.

Swarms of non-communicating robots have also been used both in simulation and in an experimental arena to investigate collective foraging, "consumption" and "grazing" (Balch and Arkin, 1994) tasks. During foraging, items needed to be found and brought back to the base. During consumption, robots arrived to item deposits and depleted them immediately. Grazing involved covering as much area as possible by moving in a pattern consisting of straight lines, similarly to a lawn mower. In all of the tasks, introducing more robots into the swarm sped up the work, especially when there were more item deposits in the arena. However, due to congestion, the positive effect of a large swarm size was only observed until a certain critical number of robots was reached.

Other authors (e.g., Sugawara and Watanabe, 2002; Labella et al., 2006; Valdastri et al., 2006) demonstrated a similar negative effect of large swarm size on the performance of robots.

The negative effect of congestion in swarms without communication has been successfully managed by using the "bucket-brigading" strategy (e.g., Drogoul and Ferber, 1993; Shell and Mataric, 2006; Lein and Vaughan, 2009; Pini et al., 2013). In a swarm that utilises bucket-brigading, each robot has a designated working area that represents a certain portion of the foraging arena. Robots can only move within their working areas and pass a collected item to a neighbour, until the item reaches a designated drop-off location. In extended versions of this strategy, the size of a robot's working area can be adapted autonomously to improve foraging efficiency (Pini et al., 2013) or to give a swarm the ability to follow mobile deposits (Lein and Vaughan, 2009).

2.3.2 Swarms with local communication

Some robot swarms rely on the ability of individuals to communicate with each other in order to achieve a goal. In (Zahadat et al., 2013), robots could distribute themselves on a grayscale floor proportionally to a human-specified required number of robots for each colour level. The robots exchanged thresholds for remaining at a certain colour with each other, and collectively achieved the desired distribution.

A comparative study of robot state and goal communication in the context of collective foraging was investigated by (Balch and Arkin, 1994). During state communication, similarly to behavioural matching (see Section 2.1.2), a robot could query another robot's state and followed it if it was a forager, i.e. if it discovered an item patch. In goal communication, a robot could acquire the exact location of an item patch from another robot and navigate directly towards it. The experiment showed that any type of communication improved foraging performance compared to a strategy with no communication, but that goal communication offered little benefit over state communication.

The performance of swarms with local, peer-to-peer communication and of "global sensing" swarms were compared for a warehouse maintenance mission, where worksites appeared and needed to be serviced, in both simulation and real world experiments (Sarker and Dahl, 2011). In the peer-to-peer swarm, robots communicated the location of discovered worksites, the worksite urgencies and the number of robots required to perform the work to any surrounding neighbours. In the global-sensing swarm, worksite information was communicated by the robots to the server, making it accessible to all members of the swarm at the same time. The swarm performance was relatively low in large swarms that used global sensing due to interference between robots that tried to access the same worksite at the same time, as well as due to frequent worksite-switching behaviour that caused the robots to travel large distances and thus perform less work.

On the other hand, large swarms that used local communication demonstrated more effective distribution among worksites.

2.3.3 Bee-inspired swarms

Bee-inspired robots also utilise local peer-to-peer communication, but, similarly to bees, they only exchange information about worksites while they are in the base. Bee-inspired swarms generally perform better than swarms with no communication in foraging missions where items that need to be collected are grouped in patches (Krieger and Billeter, 2000; Bailis et al., 2010; Pitonakova et al., 2014).

Bee-inspired robots often rely on their local information about the position of items, estimated using *odometry* (e.g., in Alers et al., 2011; Gutiérrez et al., 2010; Ducatelle et al., 2014). It is assumed that no global point of reference exists and that the robots thus need to store a relative vector towards a discovered item. The vector is updated in small steps, while the robot travels. The accuracy of odometry is thus highly dependent on the accuracy of a robot's wheel motion sensors and on the amount of wheel slippage. Odometry error grows with travelled distance, especially when the robot makes a lot of turns (Chong and Kleeman, 1997; Borenstein, 1998; Martinelli, 2002). When odometry-based information about worksite location is exchanged between robots, the receiver of the information needs to translate the local vector of the sender to its own local coordinate system (Gutiérrez et al., 2010).

There are a number of approaches to how local information about deposit locations can be shared between robots. For example, a successful forager can approach a waiting robot in the base and instruct it to follow to a discovered item deposit (Krieger and Billeter, 2000). In a centralised implementation, position of a patch can be stored in the base by successful foragers, making it available to other robots that arrive to the base later (Alers et al., 2011; Hecker et al., 2012). Alternatively, successful foragers can share deposit locations with other robots currently waiting in the base (Lemmens et al., 2008; Lee and Ahn, 2011). The swarm size plays an important role, since a large number of waiting robots can cause congestion in the base and decrease the swarm performance (Lee and Ahn, 2011).

2.3.4 Ant-inspired swarms

Ant-inspired robots utilise stigmergy in order to exchange information about where items of interest are located during foraging. In a simulated robotic experiment, robots dropped "crumbs" on the ground in order to create trails from the base towards food (Drogoul and Ferber, 1993). While trail following improved the swarm performance, crumbs also needed to be removed by robots at a suitable rate in order to prevent the

robots from traveling to locations where items were already collected. In a different simulation environment, the use of pheromone and its evaporation and dispersion has been simulated (Fujisawa et al., 2014).

In the real world, stigmergy was facilitated by letting robots emit LED light on a phosphorescent floor in order to create glowing paths (Mayet et al., 2010), as well as by allowing robots to deposit alcohol trails and to use chemical sensors to follow them (Russell, 1999; Fujisawa et al., 2014). In another experimental setup, a centralised server store virtual pheromone deposited by robots and a projector was used to display the trails on the ground, allowing the robots to follow them using visual sensors (Kazama et al., 2005; Garnier et al., 2007).

Some researchers avoided the difficulty of using pheromone-like substances, that require a specific arena setup, and programmed designated stationary robots to receive and store virtual pheromone that other, working, robots could deposit and read (e.g., Hoff et al., 2013; Ducatelle et al., 2011). In other experiments, the whole swarm was used as a medium that held pheromone paths, along which virtual ants traveled, in order to establish the shortest path to a resource (Campo et al., 2010).

2.3.5 Robot swarms in dynamic environments

In dynamic environments, worksite locations or priorities change over time. These environments deserve a special attention, since they resemble real-world tasks, such as pick up and delivery of items, or disaster response, more than environments where worksites do not change at all. In order to be able to perform missions in dynamic environments, swarms need to be able to respond promptly and appropriately to unpredictable circumstances.

A typical kind of dynamic environment explored in the swarm robotic literature includes worksites that spontaneously appear and require robots to discover and perform work on them. Worksites might also have different priorities or require a different number of robots to get involved. *Vacancy chain scheduling* (Dahl et al., 2009) is a distributed multi-robot task allocation algorithm, where robots maintain a set of discovered worksites in their memory and complete them by priority, without communicating with each other. Dahl et al. showed that the VCS algorithm can achieve optimal allocation of robots to worksites when new worksites appear and when robots are removed from the swarm, but that the algorithm requires the robots to have a reliable knowledge of the environment. In other approaches to the task allocation problem, robots communicated worksite information to each other (Gerkey and Mataric, 2003; Wawerla and Vaughan, 2010; Sarker and Dahl, 2011) or to a centralised planner (Stiefelhagen et al., 2004; Wawerla and Vaughan, 2010) and committed to worksites according to their rewards and

priorities. Auction-based approaches, where robots bid for worksites using a centralised facilitating server, have also been explored (Mataric et al., 2003; Thiel et al., 2009).

In similar missions, robot swarms were required to maximise their energy efficiency by adjusting the number of foragers based on the number of available items in the environment. A solution where robots communicated their states to each other and adjusted their foraging probability based on their own success and the success of other returning foragers, delivered a nearly optimal performance in large swarms (Dai, 2009). In other experiments, robots measured their own foraging performance and the performance of others and adapted their individual control parameters accordingly (e.g., Campo and Dorigo, 2007; Liu et al., 2007a).

Finally, some missions require robots to collect different types of items proportionally to the item type abundance that changes over time. Decentralised control algorithms, where robots communicated discovered items to nearby swarm members, were used successfully to solve this task in simulation (e.g., Jones and Mataric, 2003; Schmickl et al., 2007). The experiments pointed out the importance of achieving the correct balance between information acquisition (i.e., scouting) and information sharing (i.e., recruitment) in decentralised swarms operating in dynamic environments.

2.3.6 Summary and future directions

Sharing of information between robots about where work is located can improve swarm performance when items of interest are found in patches (Balch and Arkin, 1994; Krieger and Billeter, 2000; Bailis et al., 2010). An important determinant of swarm performance is the swarm size, since interference between robots can cause congestion and prevent the robots from from working (Sugawara and Watanabe, 2002; Labella et al., 2006; Valdastri et al., 2006). Interference is especially problematic when "global sensing" is used, i.e., when the location of discovered worksites is uploaded by robots to a server and shared with the whole swarm (Sarker and Dahl, 2011). On the other hand, local, peer-to-peer recruitment is more scalable and it is used more commonly than global sensing (e.g., in Balch and Arkin, 1994; Zahadat et al., 2013). A special type of peer-to-peer communication strategy is bee-inspired recruitment, where robots meet in the base in order to exchange information (e.g., in Krieger and Billeter, 2000; Lemmens et al., 2008; Lee and Ahn, 2011), or upload information into a device in the base that can be read by other robots later on (Alers et al., 2011; Hecker et al., 2012). In contrast, ant-inspired robots drop cues in the environment, for example by using chemical (Russell, 1999; Fujisawa et al., 2014) or light markers (Mayet et al., 2010) in order to create trails between base and worksites, helping other robots to navigate to where work is located. In other ant-inspired swarms, stationary robots store a virtual pheromone that working robots can deposit and read (Hoff et al., 2013; Ducatelle et al., 2011).

Often, no global reference point for navigation exists and robots thus need to store relative location of worksites in their own memory. The robots utilise odometry, meaning that they update a local vector pointing to a worksite every time they move (e.g., in Alers et al., 2011; Ducatelle et al., 2014). When odometry-enabled robots exchange worksite locations, the receiver of the information translates the local vector of the sender to its own local coordinate system (Gutiérrez et al., 2010).

Dynamic missions, where robots need to discover and attend to worksites that spontaneously appear in the environment, or where the swarm needs to adjust its foraging effort according to a changing availability of items in the environment, robot control strategies that achieve a balance between exploration, recruitment, and exploitation are required (Jones and Mataric, 2003; Schmickl et al., 2007). There is a range of robot control strategies capable of handing dynamic environments, including using of a fully centralised planner server that instructs robots what to do (e.g., Stiefelhagen et al., 2004; Wawerla and Vaughan, 2010), auction-based approaches (e.g., Mataric et al., 2003; Thiel et al., 2009), vacancy chain scheduling (Dahl et al., 2009) and fully decentralised algorithms that only rely on local perception and communication between robots (e.g., Gerkey and Mataric, 2003; Wawerla and Vaughan, 2010; Sarker and Dahl, 2011). It is important to point out that algorithms that rely on a central unit to plan robot actions or to facilitate robot communication are less scalable than decentralised algorithms (Mataric et al., 2003; Campbell and Wu, 2011; Fernandez-Marquez et al., 2013) and often require perfect knowledge of the environment (Dahl et al., 2009; Kernbach et al., 2012). Therefore, their use is often restricted to fully known and controllable environments, such as warehouses (e.g., Vivaldini et al., 2010; Mather and Hsieh, 2012) or hospitals (e.g., Thiel et al., 2009).

In the early nineties, is was proposed that swarms of autonomous robots could be deployed for difficult missions like solar system exploration or establishment and maintenance of underwater and lunar bases (Brooks, 1989; Brooks et al., 1990). However, the problem of how we should engineer robots, and more importantly how the robots should cooperate without human intervention, turned out to be harder than it was originally anticipated. Twenty-five years later, application of robot swarms is mostly restricted to agriculture (e.g. Cartade et al., 2012), drug delivery in hospitals (Thiel et al., 2009) or logistic tasks in warehouses (D'Andrea, 2012; Mather and Hsieh, 2012; Stiefelhagen et al., 2004; Vivaldini et al., 2010). The robots operate in well-defined environments and are so far not capable of autonomy or self-organisation. More difficult missions, in which swarms have to react to changes in the environment, have so far been mostly explored in simulation and the research is still relatively new (see Section 2.3.5). It is going to take us some time to understand swarm intelligence well enough to be able to use it in the real world.

The logistics and transportation sectors will be crucial in this endeavour in the near future. Air (e.g., Amazon Prime Air, 2016) and ground (e.g., Starship Technologies,

2016) robot delivery systems are currently being developed for delivering food and other light packages to customers. The main challenges that the industry currently faces are how to design the robots to perform work safely and efficiently, as well as how the robots should localise themselves and coordinate with each other (D'Andrea, 2014). Multi-robot coordination is the most relevant to the field of swarm robotics. Thousands of robots will need to autonomously, and in real time, decide what packages to pick up and deliver first, or how to share charging stations. Furthermore, robot engineers, software developers, business owners and governments need to find a way of addressing concerns about safety and privacy (Hansenberger, S. and Wildhaber, I., 2016; D'Andrea, 2014), especially when it comes to autonomous flying robots, i.e., drones.

The development and application of autonomous self-driving cars face similar challenges. There are unresolved concerns about user freedom and privacy, as well as about how to regulate accident liability (Boeglin, 2015). Nevertheless, companies like Nutonomy (Nutonomy, 2016), Google's Alphabet (Ziegler, C., 2015) and Uber (Griswold, A., 2016) are in a race to develop fleets of self-driving taxis, where a customer's request to be picked up and driven to a place of their choice could be autonomously fulfilled by the cars. The most attractive property of self-driving cars is their increased safety, compared to human drivers (Boeglin, 2015). Self-driving cars on the road could communicate with each other in a peer-to-peer fashion in order to avoid and mitigate crashes or to allow multiple families to safely share a road trip together.

As the scale and demand for these systems increases, their autonomy will also have to scale up. We would like to reach a stage where autonomous drones and vehicles seamlessly and safely integrate with our society. It is currently predicted that we could see autonomous cars as soon as in 2019 (Sanburn, J., 2012; Greenough, J., 2016; Bhuiyan, J., 2016) and that human drivers will be completely replaced by AI before the year 2050 (Bhuiyan, J., 2016). In terms of drone delivery systems, Amazon Prime Air is already being tested in the United States and in the European Union, although the full commercial launch of the technology is impossible before relevant legislation is established (Amazon Prime Air, 2016).

Another important sector for swarm robotics is the space industry. In the new century of computing, where processors, physical memory and batteries are becoming exponentially smaller, cheaper and more powerful, there has been a renewed interest in using robot swarms for terrain exploration, site preparation and habitat construction on the Moon and on Mars (e.g. Fong and Nourbakhsh, 2000; Huntsberger et al., 2000a,0; Vane et al., 2010), as well as for human-robot cooperation on space missions (Fong et al., 2006). Recent advances in space robotics, caused by an increasing interest in the Moon and Mars from the European (ESA Space Applications Workshop, 2014; ESA Mars Missions, 2015) and American (NASA Mars Missions, 2015) space agencies, as well as from the private sector (Google Lunar X Prize, 2015; Astrobotic, 2015), might finally push swarm robotic research beyond the laboratory. While robotic swarms could be a

good substitute for human workers in the inhospitable space, high level of uncertainty, caused by unpredictable robot failures, difficult terrain and limited energy supply are expected (Huntsberger et al., 2000a; Schenker et al., 2001). Therefore, we are starting to appreciate robustness of decentralised systems, even if it means that optimality of work performance needs to be sacrificed (Benaroya et al., 2002; Fong et al., 2006). In a similar fashion to how evolution shaped natural species for survival rather than for perfection, engineering for unpredictable environments is moving in the same direction.

However, even if we do not strive for optimality, we would still like to make sure that robot swarms will perform as best as they can. Understanding of swarm intelligence on a level that allows us to design autonomous and reliable group behaviour is therefore crucial to our success. While the environments, that we will send our robots to, might be dynamic and to some extend unpredictable, they will not be completely unknown. For example, we might know that we want a swarm to explore the terrain and collect mineral samples in a specific area of Mars. We might know how many samples we need and what kinds of minerals are more valuable to us. We might know that the minerals occur in small quantities and are randomly scattered or that the robots need to search for rare mineral veins. Or we might know that solar-powered robots will have limited time and place where they can recharge. A swarm designer should be able to use such facts when choosing how the robots should behave and how they should communicate, in order to create a robot team that fits a given mission as best as it can. Similarly as in software engineering, it should be possible to work from a set of "behavioural modules" and to construct the correct type of macro swarm behaviour through guided design of behaviour on the individual, micro, level of robots. Before we can do this, we need an abstract, context-generic understanding of swarm intelligence in embodied systems in order to identify what types of behaviour are suitable under given mission conditions.

2.4 Design patterns

A design pattern offers a flexible high-level solution to a class of well known problems, that a programmer can implement in a particular, context-specific way (Schmidt, 1995; Do et al., 2003). In object-oriented software engineering, design patterns define what roles object classes should have and how they should interact (Gamma et al., 1994; Eden et al., 1997). In agent-based software engineering, design patterns can define roles and interactions of agents, as well as the role of the environment (Tahara et al., 1999). Good patterns that use a common unambiguous language can decrease system design time, as well as improve communication between engineers (Schmidt, 1995; Brazier et al., 2002; Gardelli et al., 2007).

Swarm robotics could benefit from a catalogue of design patterns that would help engineers to design robot behaviour based on known mission parameters (Brambilla et al.,

2013; Reina et al., 2014). A number of automated methods for implementing robot swarms exist, for example evolutionary algorithms (e.g., Niv et al., 2002; Sperati et al., 2011; Ferrante et al., 2015; Doncieux et al., 2015) and reinforcement learning (e.g., Campo et al., 2010; Dahl et al., 2009; Pérez-Uribe, 2001; Yasuda et al., 2014). Additionally, we can analyse swarm-level behaviour and optimise robot algorithms, for example by creating probabilistic finite state machine models (e.g., Liu and Winfield, 2010; Mather and Hsieh, 2012). However, these methods do not provide any guidelines on how an appropriate robot algorithm should be selected (see Section 10.2 for a further discussion), and their use is thus mostly limited to parameter optimisation. For example, they cannot tell us whether a swarm should be ant- or bee-inspired.

In this section, an overview of design patterns for object-oriented software engineering and for multi-agent software systems is first given, including how the patterns are specified and how they can be applied to programming problems. The current design pattern trends in swarm robotics are then discussed and rationale for the need to perform more research in this field is given.

2.4.1 Object-oriented software engineering

Design patterns in object-oriented (OO) software engineering belong to one of the three following categories (Gamma et al., 1994):

1. *Creational*, that dictate how new instances of classes should be generated
2. *Structural*, that define where objects should reside within a program
3. *Behavioural*, that define responsibilities and interactions of objects

A strong aspect of design patterns is their modularity (Mikkonen, 1998). A single pattern rarely defines the structure of the whole program. Rather, it can define a solution to a specific problem, e.g. how to instantiate objects, how to manage data, etc. Multiple design patterns are often combined within a single program and a programmer decides how to implement them together, subject to restrictions of a specific programming language that is used.

A clear visualisation facilitates understanding and it is thus an important part of a design pattern description (Tahara et al., 1999). In OO software development, UML class diagrams are commonly used to visually represent classes and their relationships. For example, in the Abstract Factory pattern (Figure 2.1a), that belongs to the Creational category, "factories" provide instances of object to other objects. There is an abstract factory class that specifies common factory methods and subclasses can be created to deliver specific object types. Another pattern example is the Observer pattern (Figure 2.1b), which belongs to the Behavioural Category. According to this design pattern,

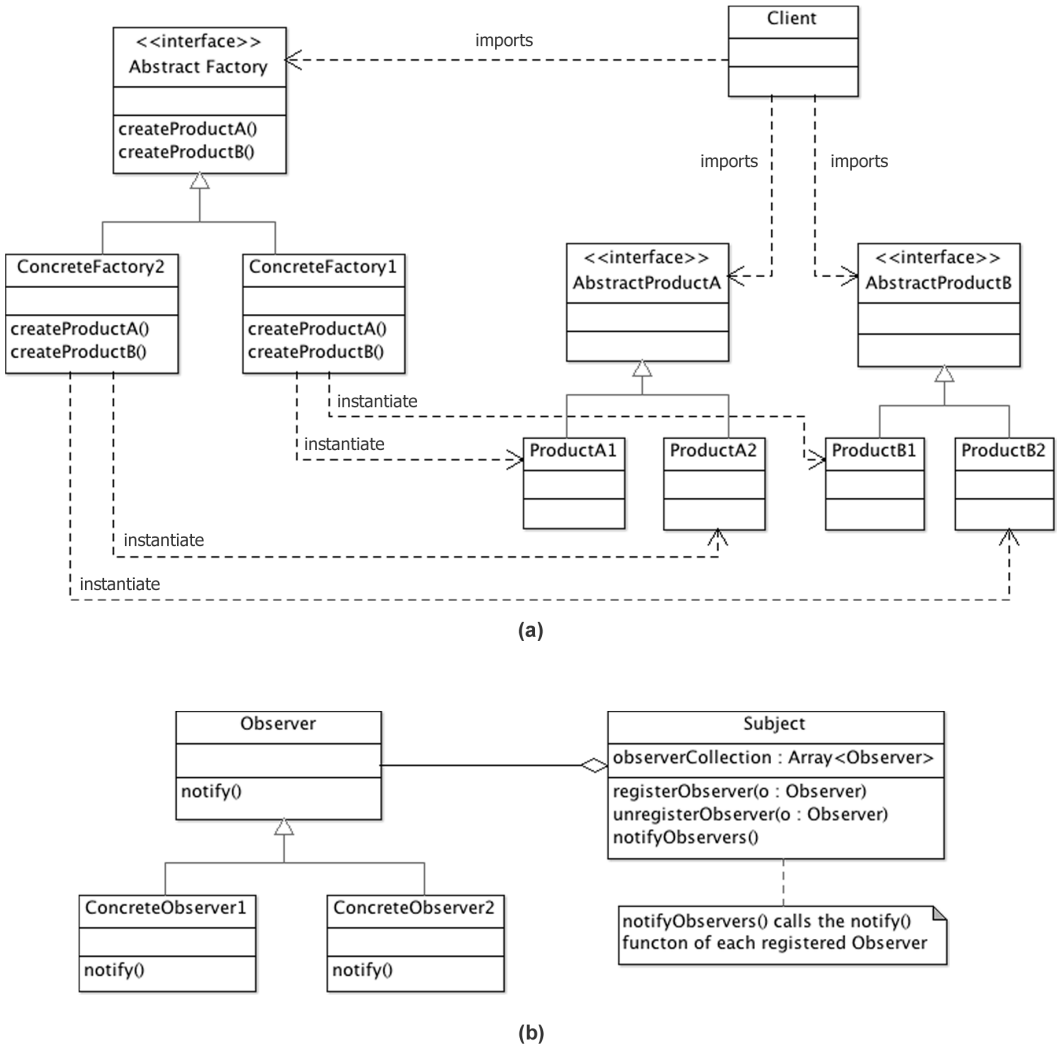


Figure 2.1: UML class diagrams of (a) the Abstract Factory and (b) the Observer design patterns.

an observer object subscribes to another object to be notified when its state changes. This pattern is often used for programming user interfaces, where keyboard and mouse events need to be handled in parallel to the main program execution.

Textual description of patterns follows a standardised format, where the pattern *title*, *problem*, *solution* and *consequences* are specified. The *problem* description covers the context within which the pattern can be used. For example, the Abstract Factory pattern specification (Gamma et al., 1994, p. 87) states that the pattern should be used when a part of a program needs to be independent of how its objects are created. A *solution* describes the elements that make up the design, as well as their roles and relationships. It is usually accompanied by a UML diagram. Finally, the *consequences* describe what trade-offs have to be made when a pattern is used. The consequences of a pattern are critical to consider when evaluating the pattern's suitability for a specific application.

For example, the Abstract Factory pattern promotes consistency among its products but makes supporting new kinds of products difficult.

2.4.2 Multi-agent systems

Multi-agent systems (MAS) take a decentralised approach to software development, that has become popular in web and big data applications. Similarly as object-oriented development before the specification of its design patterns, MAS development is facing difficulties in the design stage of projects due to the lack of common vocabulary and of a framework that could unify ad-hoc solutions and prevent duplicate effort (Deugo et al., 2001). While numerous attempts have been made to define design patterns for MAS, there is no clear consensus of how they should look like or on what level they should be applied.

Nevertheless, a number of common MAS patterns can be identified throughout the literature. The *master-slave* pattern (Aridor and Lange, 1998; Deugo et al., 2001) defines a structure where one agent delegates tasks to other agents and collects and delivers results to the user. In the *blackboard* pattern (Deugo et al., 2001), agents can share data through a common place, where the read-write access is managed by a designated Supervisor programme. Finally, the *broker* pattern (Deugo et al., 2001; Do et al., 2003) specifies that a separate programme should manage auction-like match making between agents that require services and agents that can provide them.

Authors often do not agree what system level the patterns should be defined for or how they should be categorised. For example, (Tahara et al., 1999) identified three pattern types: *macroarchitecture*, a generic platform-independent solutions, *microarchitecture*, i.e. platform-dependent agent behaviours, and *object-level*, i.e. particular OO implementation. On the other hand, (Aridor and Lange, 1998) argued that patterns should only be concerned with agent-level behaviour and proposed three pattern types: *traveling* that described how to perform agent routing, *task*, specifying agent roles, and *interaction*, that described how agents could be located and how to facilitate their interactions. Similarly, (Mikkonen, 1998) stated that patterns should define behavioural modules of agents, while (De Wolf and Holvoet, 2007) was concerned with coordination mechanisms between agents. (Do et al., 2003) distinguished two types of MAS design patterns: *peer*, that described interactions between agents and *mediation* that described techniques for intermediate agents that helped to facilitate interactions of other agents.

Some authors have recognised the importance of visualisation when describing MAS design patterns. For example, an agent-behaviour diagram (Figure 2.2a) represents a physical configuration of a system, a state transition diagram (Figure 2.2b) shows transitions between a system's actions, and an event sequence chart (Figure 2.2c) describes agent lifelines and interactions (e.g. Tahara et al., 1999). Other authors have been more

concerned with particular implementations of their patterns and have used UML class diagrams common in OO software engineering (Aridor and Lange, 1998). The UMLs have also been extended to show how information flows between objects (De Wolf and Holvoet, 2007).

Despite the differences in their categorisation and visualisation, MAS design patterns are usually described using similar attributes (Schmidt, 1995; Aridor and Lange, 1998; Deugo et al., 2001; De Wolf and Holvoet, 2007; Gardelli et al., 2007; Fernandez-Marquez et al., 2013), namely the *problem* that a pattern can solve, the *context* within which a pattern is applicable, including what *forces* play an important role in its application, *participants* and *collaborations*, i.e. the agents and they roles, and *consequences*, i.e. how a solution proposed by the pattern works and the cases when it does not work. Some

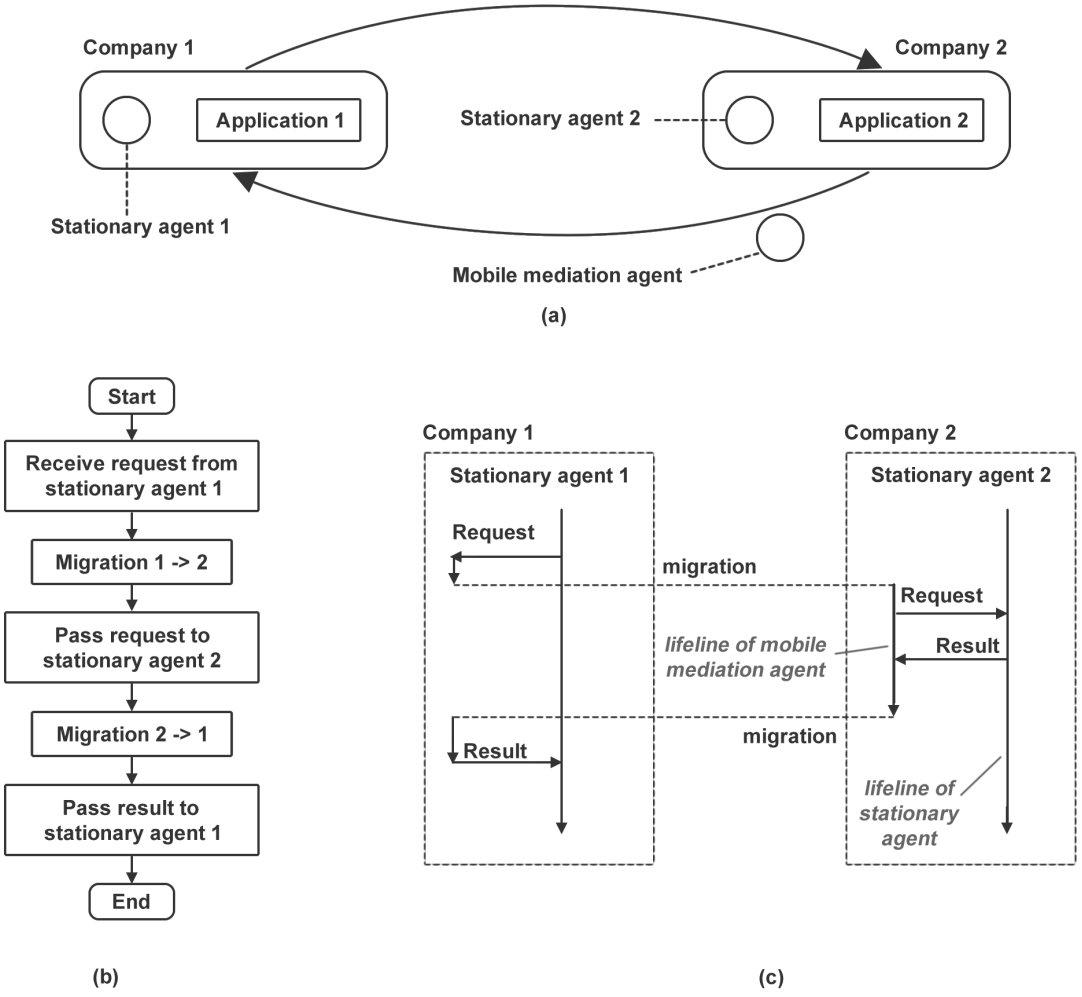


Figure 2.2: An example of (a) an agent behaviour diagram, (b) a state transition diagram and (c) an event sequence chart for the Mediation pattern, where a mobile mediator agent mediates cooperation between stationary agents (adapted from Tahara et al., 1999).

authors also provide a list of related patterns and an example source code (Schmidt, 1995; De Wolf and Holvoet, 2007).

2.4.3 Application to robotics

Even though robot swarms are a subclass of multi-agent systems, robots, unlike software agents, are embodied entities. While software agents can share data relatively easily and can travel from one host computer to another within milliseconds, robots have a limited world view are subject to physical restrictions like movement speed, collisions, component failures, energy consumption, etc. Furthermore, most of the MAS design patterns rely on the existence of a programmable encapsulating environment, i.e. a host application, where data can be stored and easily retrieved, while robots operate in the real world that is out of a programmer's control. For these reasons, traditional MAS design patterns are difficult to apply in robotics.

One of the first attempts to specify design patterns for swarm robotics came from the field of multi-robot self-assembly (Nagpal, 2004). Expanding on the idea of pattern modularity, Nagpal defined a set of behavioural primitives and categorised them into the *local* and *global* levels. Local-level primitives described how robots could interact and share data with each other. For example, *morphogen gradients* could be used for localisation within a robot collective and *quorum sensing* was useful for counting whether a specific number of robots is nearby. Global-level primitives described how local primitives could be combined together. For example, *cell differentiation* could be achieved by combining *morphogen gradients* and *labour division*.

The environment plays an important role for robots that perform work, since the robots modify the environment and need to adapt to it. (Gardelli et al., 2007), who was interested in ant-inspired robot foraging, took the agent-environment interactions into the account when they identified patterns including *collective sorting*, *pheromone evaporation*, *agent aggregation* and *diffusion*. However, the patterns were very application-specific and the authors did not provide any discussion about their consequences or general lessons learned. (Fernandez-Marquez et al., 2013) extended Gardelli et al.'s work and identified patterns on three levels: *basic* that more or less mirrored the patterns of (Gardelli et al., 2007), *composed* that could be created by combining the basic patterns, e.g. use of a *digital pheromone*, and *high-level* patterns that defined swarm-level behaviours like *flocking* or *foraging*.

Despite of the effort of numerous authors, a design pattern catalogue for swarm robotics does not currently exist. Similarly to OO software development, swarm engineering requires a common language and visualisation methods that could facilitate creation and discussion of patterns. Secondly, pattern creation requires practical experience (Schmidt, 1995), but the field of swarm robotics is relatively new and its applications

are currently limited (see Section 2.3.6). Therefore, simulations, that are a useful tool for exploring a large number of robot behaviours and their parameters, could play a crucial role in the development of design patterns for robot swarms.

Chapter 3

Methods

The aim of this thesis is to provide a catalogue of design patterns for swarm robotics. The design patterns should be implementation-generic "modules" of robot behaviour that can be combined together into a robot control strategy suitable for specific mission requirements. The robot swarms should be able to operate autonomously and without a central controller on tasks such as item retrieval and delivery, equipment repair and servicing, or emergency response.

In order to create design patterns, a considerably large number of use cases needs to be investigated, so that the performance and properties of various robot control strategies can be thoroughly analysed and compared (Schmidt, 1995). Furthermore, a framework is required that can provide a theoretical terminology within which behaviour of swarms can be understood and talked about.

In order to collect a sufficiently large amount of experimental data, 120 different experiments were run, each with three swarm types (solitary robots, local broadcasters and bee swarms, see Sections 3.1.2– 3.1.4) and three swarm sizes (10, 25 and 50 robots). Every experiment consisted of 50 independent simulation runs, resulting in total of 54000 simulations.

In order to make sense of the data, the *Information-Cost-Reward* (ICR) framework is developed (see Section 4.4.5). The framework provides an abstract, information-based, account of the experimental results and it is used in Chapters 4–8 to form hypotheses about expected swarm performance and to explain the results from various robot missions. Finally, the ICR framework provides terminology for describing swarm design patterns in Chapter 9.

Missions where robots need to search for a number, N_W, of worksites in the environment in order to obtain reward from them are explored. There are two types of scenarios (Figure 3.1):

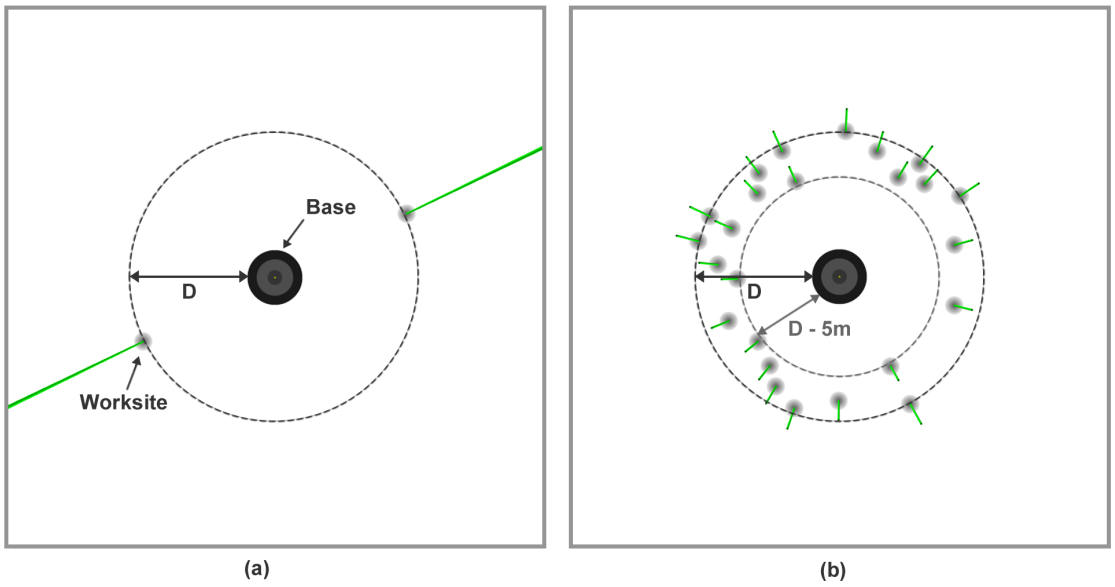


Figure 3.1: ARGoS simulation screenshot of the experimental arena containing a base in the centre and worksites in the (a) Heap2 and (b) Scatter25 scenarios with worksite distance $D = 13$m. The worksites are represented as cylinders, with their heights corresponding to their respective volumes, and have colour gradients with radius $r_C = 1$m around them to guide navigation of nearby robots (see Section 3.1.1).

- **Heap$N$:** $N_W \epsilon \{1, 2, 4\}$ high-volume worksites distributed evenly around the base at a distance D from the base edge.
- **Scatter25**: $N_W = 25$ worksites randomly distributed between distance D and $D - 5$m from the base edge.

The total amount of resource in each scenario is set to 100 and the amount of reward per worksite, $V = 100/N_W$. For example, each worksite in a Heap2 scenario has $V = 50$, while worksites in the Scatter25 scenario have $V = 4$. Each scenario is investigated with five different values for worksite distance, $D \in \{5, 9, 13, 17, 21\}m$, from the base edge.

Two basic types of robot tasks are investigated in each scenario:

- **Consumption**: worksites can be depleted by nearby robots at a *reward gain rate* $\rho = 1/400$ units per second. The swarm's total reward is increased in each simulation time step when robots are near worksites. This type of task is analogous to the "consume" task explored by (Balch and Arkin, 1994) or the "job completion" problem on a manufacturing floor (Gerkey and Mataric, 2003; Dahl et al., 2009; Sarker and Dahl, 2011).
- **Collection**: worksites represent resource deposits, where a robot can collect a maximum of 1 unit of volume of resource at a time. A robot takes 1 second to load

the resource and returns to the base in order to unload it. Reward is obtained in the base during unloading at the rate of $\rho = 1$ units per second. Similar tasks, where items of interested appear in patches and needed to be collected, were explored e.g. in (Krieger and Billeter, 2000; Lee and Ahn, 2011; Lemmens et al., 2008; Ducatelle et al., 2011).

Worksites in both tasks are depleted faster when more robots work on them at the same time. The loading rates for the two tasks are set so that worksites take on average a similar time to deplete during both consumption and collection tasks, and when different experimental environments are considered.

Additionally, there are three variations of each task:

- **Static**: worksite locations are determined at the beginning of a simulation run and remain constant.
- **Slow dynamic**: worksite locations change periodically each T_C minutes, creating environments where continuous exploration of the work arena needs to be balanced with worksite exploitation. A new location is chosen randomly for each worksite according to the scenario type at the end of each T_C interval. For example, in a Heap2 scenario with $D = 5$m, two new locations are generated, each 5m away from the base edge. Worksites are replenished after each change. The value of T_C is adjusted for each swarm size, based on the average performance measured in the static task (see Chapter 6).
- **Fast dynamic**: The change interval for each swarm size is set to half of the T_C used in the slow dynamic task, creating highly dynamic environments, where exploration is very important.

Each experiment is performed with three types of homogeneous robot swarms, that represent three different robot control strategies (see Sections 3.1.2 – 3.1.4):

- **Solitary swarm**: robots do not communicate about worksite locations with each other
- **Local broadcasters**: robots broadcast information about a worksite that they are currently working on to other robots that are nearby
- **Bee swarm**: robots meet in the base in order to exchange information about worksites

Additionally, two biologically inspired add-on control strategies are investigated for each basic control strategy in Chapters 7 and 8:

- **Opportunism**: robots continuously seek information about new worksites, in a manner defined by the behavioural rules of their particular basic control strategy. Robots opportunistically choose to subscribe to worksites that have a better utility, i.e., those that are either closer to the base or have a larger volume.
- **Anticipation**: robots anticipate low returns from worksites that they are subscribed to by continuously evaluating the utilities of their worksites. Robots abandon their worksites with a probability that increases with decreasing worksite utility, i.e., as the worksites get depleted.

Experiments with these add-on behaviours are performed in the dynamic collection and consumption tasks, using all three swarm types and all three swarm sizes, resulting in data from additional 72000 experimental runs.

In the following sections, the simulation environment and the basic control strategies of robots are described, followed by a summary of the analysis methods used throughout this thesis.

3.1 Implementation

3.1.1 Simulation environment

All the experiments reported here are performed in the ARGoS simulation environment (Pinciroli et al., 2012) using MarXbot robots (Bonani et al., 2010). ARGoS is a C++ environment with a realistic 3D physics engine that was specifically designed for program compatibility with MarXbots (also called FootBots) and e-Puck robots. By using this type of simulation environment, physical interactions between robots, as well as between robots and the environment can be modelled. The physical aspect of simulation is important, since interference between robots can significantly affect swarm performance. For example, congestion prevents robots from working when multiple robots try to access the same worksite at the same time (see Chapter 4).

The simulation takes place in continuous space and updates itself 10 times per second. The experimental arena is 50m×50m large and contains a centrally located circular base surrounded by worksites (Figures 3.1 and 3.2). A similar setup has been previously used in simulated (e.g., Balch and Arkin, 1994; Campo and Dorigo, 2007), as well as real world (e.g., Labella et al., 2004; Gutiérrez et al., 2010) experiments.

The base has a radius of three metres and it is divided into two sections: an interior circular *recruitment area* and an annular *unloading area* around it (Figure 3.2). A light source is placed above the middle of the base that the robots can use as a reference

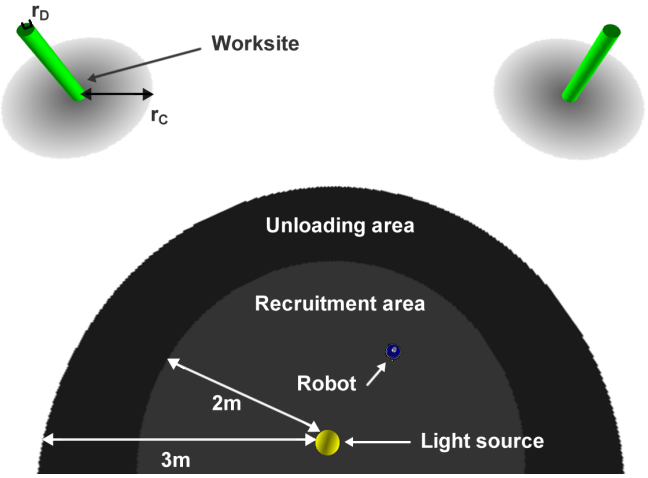


Figure 3.2: ARGoS simulation screenshot of a base and nearby worksites. The base consists of a circular recruitment area with a radius of 2m, utilised by a particular robot controller for exchanging information between robots, and an unloading area that forms a ring around the recruitment area, where returning workers unload collected material during the collection task. There is a light source above the centre of the base to guide robot navigation. The whole base has a radius of 3m. A robot, with radius of 8.5cm is located in the recruitment area. Each worksite has a colour gradient around it to allow nearby robots to navigate towards it.

for navigation towards and away from the centre of the base (as in, e.g., Krieger and Billeter, 2000; Ferrante and Duéñez-Guzmán, 2013; Pini et al., 2013).

Cylindrical worksites with radius $r_D = 0.1$m are placed outside of the base, each containing $V = 100/N_W$ units of resource, where N_W is the total number of worksites. In order to enable robots close to a worksite to move towards it, a colour gradient with radius $r_C = 1$m is centred on the floor around each worksite. In a real-world experiment, navigation based on the colour gradient could be replaced by visual-based navigation, for instance.

The simulated MarXbots (Bonani et al., 2010) are circular robots with a radius of 8.5cm. They are equipped with four colour sensors pointed to the ground, a ring of 24 infrared proximity sensors used for collision avoidance, a light sensor used for navigation towards the base, a range and bearing module used for localisation of other robots and for communication, differential steering sensors on each wheel utilised for odometry and a ring of eight colour LEDs used for debugging.

The robots are modelled using the following assumptions:

- The swarm is homogeneous, meaning that each robot executes the same control algorithm. Variations in hardware are not modelled.
- The robots are differentially steered and can reach a maximum speed of 5cm/s.

- The maximum range at which the robots can detect objects via the proximity sensors is 0.3m.
- The light source above the base can be detected from any distance.
- The worksite distance, D, is known by the robots upfront, meaning that they consider the working arena to be circular, with a radius of Dm away from the base. The robots do not travel further than Dm away from the base while they explore the environment. However, they do not know where exactly the worksites are located.
- The robots are able to utilise a differential steering sensor for odometry (see Section 3.1.2).
- Minor differential steering sensor noise and wheel slippage are modelled. Consequently, minor odometry errors may occur, especially when robots perform many turns while avoiding each other.
- The swarm is fully decentralised and any communication between the robots happens locally, using the range and bearing module with a signal range of approximately 5m.
- The range and bearing module is based on line of sight, and hence intermittent ranging and communication problems are possible, although they do not have a significant effect.
- Information is transmitted between the robots in packets, where a single packet is transmitted in each simulation step. The robots need to maintain a communication link for a sufficient amount of time in order to transmit information that consists of several packets (e.g., a relative vector pointing to a worksite that consists of a length and an angle).
- Infra red sensor noise, light sensor noise and range and bearing module noise are not modelled. It is assumed that there is no package loss during communication.

The robots are modelled as finite-state machines. Each control strategy, described below, has a number of parameters associated with it (see Table 3.1). The particular parameter

Parameter description	**Symbol**	**Value**
Neighbourhood search time	T_N	180 seconds
Neighbourhood search radius	r_N	3 metres
Scouting probability (bee swarm)	$p(S)$	10^{-3}
Maximum scouting time (bee swarm)	T_S	18 minutes
Recruitment time (bee swarm)	T_R	120 seconds

Table 3.1: Parameter values used by various robot control strategies described in Sections 3.1.2 – 3.1.4. See Appendix A for more details.

values were selected in order to maximise performance of the control strategies in the static collection and consumption tasks. See Appendix A for more details on parameter optimisation.

3.1.2 Solitary swarm

The finite state machine representation of a solitary robot is shown in Figure 3.3. A robot starts in a random orientation and at a random position in the base as a *scout* and leaves the base immediately in order to search for worksites using Lévy movement (Reynolds and Rhodes, 2009). During Lévy exploration, a robot makes random turns drawn from a Lévy distribution function of the GNU Scientific Library (GNU Scientific Library, 2001), with the function parameters $c = 1.9$ and $\alpha = 2.0$. Small values are drawn from the distribution more frequently, meaning that scouting comprises of frequent movements along approximately straight lines and of less frequent large turns.

While outside the base, a robot updates its estimation of the relative position of the base using path integration based on odometry at each time step (as in, e.g., Alers et al., 2011; Gutiérrez et al., 2010; Ducatelle et al., 2014).

When a worksite is discovered, the robot navigates to it using the colour gradient and starts working. During the consumption task, work is performed by extracting worksite volume at loading rate of $\rho = 1/400$ units per second. When the worksite is depleted, the robot resumes scouting.

During the collection task, the robot loads one unit of volume and uses phototaxis in order to return to the base and unload the resource[1]. The robot keeps track of its relative position to the worksite by using odometry and returns to it if it did not previously deplete it, otherwise it resumes scouting.

If a robot reaches a worksite location but the worksite cannot be found, the robot performs *neighbourhood search* that lasts $T_N = 180$ seconds, in order to account for a

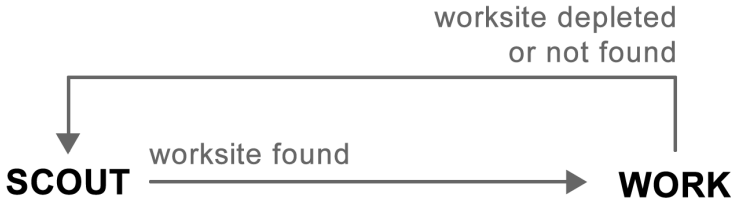


Figure 3.3: Finite state machine representation of a solitary robot

[1]The unloaded resource has no physical representation in the simulation, i.e., it is assumed that it "disappears" from the unloading area as soon as a robot unloads it. We have explored the effect of congestion, caused by accumulated resource in the base, in Pitonakova, L., Crowder R. & Bullock, S. (2016). "Task allocation in foraging robot swarms: The Role of Information Sharing", see Appendix J.

possible odometry error. During neighbourhood search, the robot moves randomly in a circular area with a radius of $r_N = 3$m around the expected worksite location. If the search is unsuccessful, the robot resumes scouting.

3.1.3 Local broadcasters

The behaviour of local broadcasters is similar to that of solitary robots (Figure 3.4). A scout searches for worksites in the environment and becomes a worker when its search is successful. Additionally, a worker broadcasts the worksite location to all robots within its communication range.

A scout that picks up the signal sends an acknowledgement to the broadcaster and a temporary peer-to-peer wireless connection is formed between the robots. The broadcaster then sends the relative position of the worksite to the receiver and the receiver translates it into its own local coordinate system, taking into account its own relative orientation towards the broadcaster (Gutiérrez et al., 2010). When the communication is completed, the recruited scout becomes a worker.

Note that in the collection task, broadcasting is only turned on while a robot is near a worksite and it is not active during the trip between the worksite and the base.

Figure 3.4: Finite state machine representation of a local broadcaster robot

3.1.4 Bee swarm

The finite state machine representation of a bee swarm robot is depicted in Figure 3.5. Inspired by the behaviour of bees, that exchange information in the nest, a successful bee swarm scout returns to the base in order to become a *recruiter* and inform other robots about a worksite that it found. A recruiter moves across the recruitment area in the base for $T_R = 120$ seconds, while advertising its worksite location to all *observers* within the communication range[2].

Observers are unsuccessful scouts that are waiting in the base for information. They move randomly in the recruitment area and avoid traveling into the unloading area. An

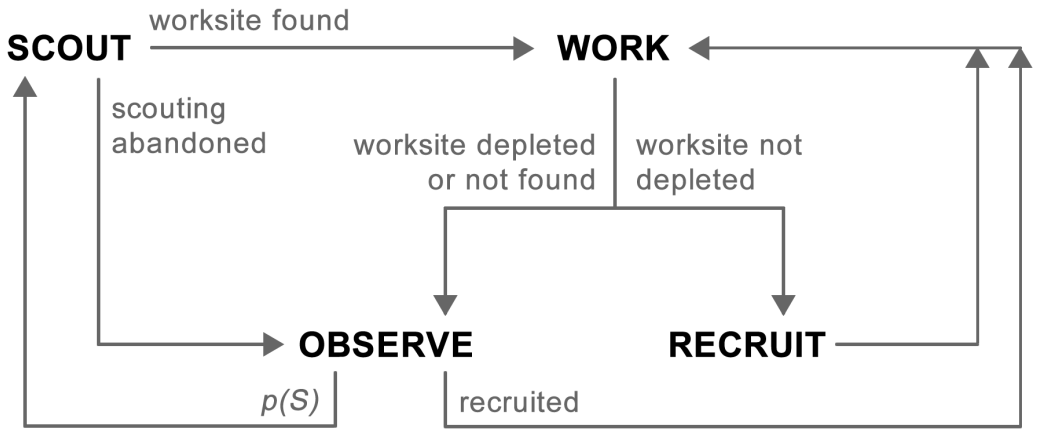


Figure 3.5: Finite state machine representation of a bee swarm robot

observer can spontaneously become a scout with the scouting probability $p(S) = 10^{-3}$ at each time step.

Similarly as it is the case for local broadcasters, a recruiter and an observer form a temporary wireless peer-to-peer connection when the recruiter sends information to the observer. A recruited robot leaves the base and heads towards the advertised worksite. The recruiter returns to its worksite after the recruitment time, T_R, expires. In the consumption task, each robot recruits only once for a specific worksite, i.e., after it initially discovers it as a scout. During the collection task, a robot recruits every time after it unloads resource in the base.

An unsuccessful scout, that cannot find a worksite within $T_S = 18$ minutes, returns to the base in order to become an observer, opening an opportunity for itself to learn about a worksite from another robot. Similarly, robots that reach a worksite location recorded in their memory but cannot find the worksite during the neighbourhood search, return to the base as observers.

3.2 Analysis

3.2.1 Performance analysis

In static tasks, where worksite locations are determined at the beginning of a simulation run, *task completion time* is used as the main swarm performance measure. A task is considered completed when all worksites in the working arena have been depleted, i.e., when all resource has been extracted from the environment.

[2]Since the communication range of robots is larger than the diameter of the recruitment area, a recruiter can reach any observer in the base, provided that other robots are not obstructing the line of sight between them.

In dynamic tasks, where worksite locations change over time, the *total reward* that swarms are able to obtain from the environment is measured. The experiments are setup so that a higher total reward is received when a swarm is better able to respond to environmental changes.

Each performance metric is based on 50 independent simulation runs with different random seeds.

When swarm performance is considered, a control strategy that performs significantly better than others is referred to as the *winning strategy*. In static environments, a winning strategy is that which has the *lowest* completion time. In dynamic environments, a winning strategy is that which has the *highest* total reward. On some occasions, there may be more than one winning strategy, provided that the differences between them are not statistically significant but that at least one of them is significantly better than any other remaining strategies. When there are no statistically significant differences between any control strategies in a given experiment, all are considered to be winning strategies.

Statistical significance is determined by using the Tukey's honest significant difference (HSD) test (Tukey, 1949) in conjunction with ANOVA, with statistical significance level $p = 0.01$. Using ANOVA, Tukey's HSD test compares each pair combination from a list of distributions, i.e., in this case, each pair of control strategies, and determines whether there are significant differences between the means of each distribution pair.

3.2.2 Information flow and cost analysis

A number of metrics that describe the information flow in swarms, as well as costs that robots incur in order to turn information into reward, are introduced in Chapter 4. Since explanation of these metrics requires a basic understanding of how the swarms behave and perform, the chapter starts by performance analysis of the three basic control strategies (solitary robots, local broadcasters, and bee swarms), using data from the static consumption task. In Section 4.2, two information flow metrics, *scouting efficiency* and *information gain rate*, are introduced. In Section 4.3, the tendency of robots to incur various costs is explored.

The relationships between characteristics of a robot control strategy (i.e., its information flow and tendency to incur costs) and the characteristics of the environment (i.e, the nature of the task and spatial structure of the work arena) form the basis of the Information-Cost-Reward framework, described in Section 4.4.

3.2.3 Visualisation

Three main visualisation methods are used throughout this thesis: box plots, time series graphs and matrix plots.

Box plots (e.g., Figure 3.6) are utilised for comparing performance or characteristics of multiple control strategies. A middle horizontal line of a box plot represents a median value of a set of results collected from 50 independent runs. The line is surrounded by a box, representing the inter-quartile range or "middle fifty" of the result set, and whiskers representing the "middle 97", with outliers outside this range shown as plus signs. When annotating axes, the letter "k" is used as a shortcut for "1000". For example, the number 5,000 is written as "5k" on the y-axis of Figure 3.6.

Time series graphs show how variables change over time (e.g,. Figure 3.7). Each data point in a time series represents a median value of a set of results, while whiskers and outliers have the same role as those of box plots.

Finally, matrix plots (e.g., Figure 3.8) are utilised in Chapters 4 – 6 to show winning strategies in different environments. The x-axis represents a scenario type, while the y axis represents worksite distance. The three explored control strategies are represented by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). A cell in a matrix plot represents a single environment (i.e., a particular combination of scenario type and worksite distance) and has a colour of a winning strategy in that environment. Multiple colours in the same cell indicate multiple winning strategies

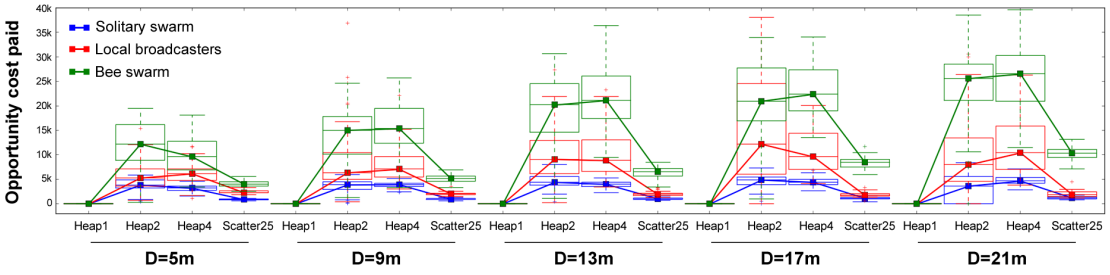


Figure 3.6: Example of a box plot graph.

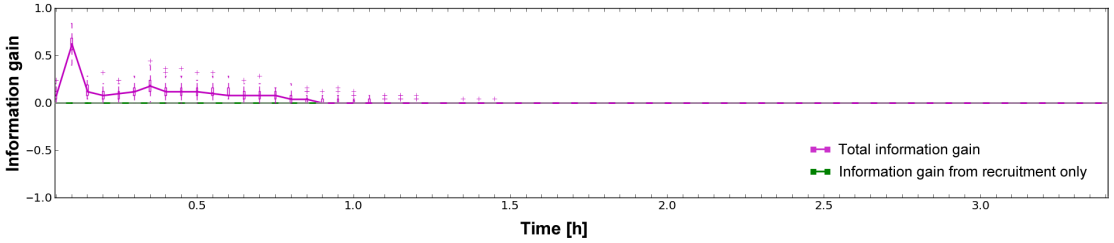


Figure 3.7: Example of a time series graph.

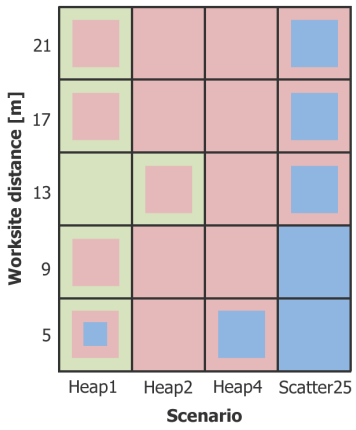


Figure 3.8: Example of a matrix plot that shows winning strategies. The strategies are identified by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). The bottom right cells indicate that solitary swarm is the winning control strategy in Scatter25 scenarios with $D \leq 9$m. The top left cells show that local broadcasters and bee swarms share a similar performance, better than that of solitary swarms, in Heap1 scenarios when $D \geq 17$m. The bottom left cell indicates that all control strategies perform similarly in Heap1 scenario when $D = 5$m.

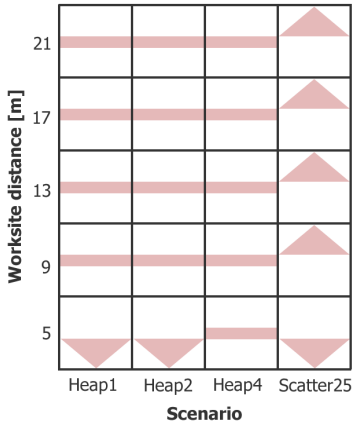


Figure 3.9: Example of a matrix plot that shows an effect of an add-on behaviour. The bottom cells indicate that the add-on behaviour impairs the swarm performance in Heap1, Heap2 and Scatter25 scenarios when $D = 5$m. The cells on the right show that the swarm performance improves in Scatter25 scenarios when $D \geq 9$m. The swarm performance is similar with and without the add-on behaviour in other scenarios.

Matrix plots are also used in Chapters 7 and 8 in order to visualise whether a particular add-on behaviour (see beginning of this chapter) improved the performance of a basic control strategy (e.g, Figure 3.9). In this case, a downward-facing arrow is used when an add-on behaviour decreased the performance, an upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on

performance. A single colour that represents a basic control strategy, that the add-on behaviour extends, is used for all symbols on a single plot.

Chapter 4

Basic concepts: Consumption in static environments

The aim of this chapter is to introduce the basic concepts and analysis methods used throughout this thesis, as well as to define the *Information-Cost-Reward (ICR) framework*. The ICR framework explains how the way in which a swarm gains and utilises information relates to its ability to obtain reward, given the particular structure of the swarm's task and of the environment. The framework is used to form hypotheses and to analyse the performance of swarms with various control strategies in Chapters 4 – 8. It also provides a context and a terminology used for describing design patterns for robot swarms in Chapter 9.

There are two robot tasks explored in this thesis: the *consumption* task and the *collection* task. In the *consumption* task, robots search for worksites in the environment and obtain reward by staying close to worksites while gradually depleting them. In the *collection* task, robots extract small packets of resource from worksites and deliver them to the base. They thus have to make a number of foraging trips in order to deplete a worksite. In both tasks, the locations of the worksites are initially unknown and must be discovered by searching the environment.

In this chapter, data obtained from the consumption task in static environments is considered. Because the environments explored here are static, i.e., worksite locations are determined at the beginning of an experimental run and remain constant during the run, performance of the swarms is only affected by the behaviour of robots, rather than by changes in the environment external to the swarms. The collection task in static environments is investigated in Chapter 5. Both tasks in dynamic environments are explored in Chapter 6.

In the first part of this chapter, performance of the three types of swarm (solitary, local broadcasters, and bee swarms; introduced in Section 3.1) is evaluated. Following the

performance analysis, information flow, i.e., the way in which robots gain and share information, is analysed. The term "information flow" is used here to describe the way in which the number of informed robots, that know where worksites are located, changes over time. It is shown that there is a discrepancy between the amount of information about worksites that a swarm has and the amount of reward that it is receiving at a given point in time. This discrepancy can be expressed as a sum of various *costs* that the swarm has to pay in order to transform information into reward. The costs described here are unitless and are related to the amount of reward that the swarm is loosing by not utilising information efficiently. The nature and the volume of the incurred costs are dependent on the structure of the swarm's environment, as well as on the way in which robots obtain and share information. The ICR framework ties together various environmental and swarm characteristics by portraying a swarm as a single cognitive entity that searches for information, utilises it, and in turn changes the environment, given the constraints of space and time.

4.1 Swarm performance

In this section, the performance of swarms is evaluated in terms of the time it takes each swarm to discover and consume all reward from the environment. A total of 20 environments are explored: four scenario types (Heap1, Heap2, Heap4 and Scatter25), each with worksite distance $D \in \{5, 9, 13, 17, 21\}$m from the base.

4.1.1 Solitary swarms

The performance of solitary swarms in the consumption task depended on the number of worksites, N_W, the worksite distance from the base, D, and the number of robots, N_R. The performance was better when there were more worksites in the environment, as it was more probable for a robot to find resource (Figure 4.1). Recall that the total amount of resource in the environment was held constant when N_W was varied by scaling the

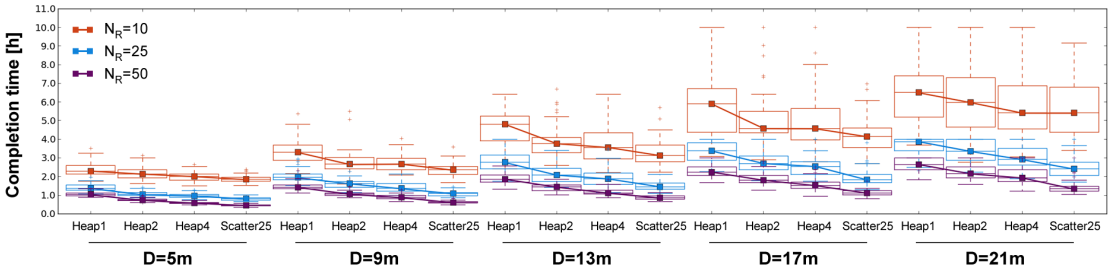


Figure 4.1: The static consumption task completion time of solitary swarms. The task completion time was shorter when worksites were numerous or when they were close to the base.

amount of resource per worksite. The increased performance in scenarios with a higher N_W was thus a result of better ability of robots to find and exploit worksites, and not a result of more abundant resource in the environment. This trend was stronger when swarms were large, since the work arena was more densely populated by robots and worksites were thus discovered faster. For example, when $D = 5$m, the task completion time in the Scatter25, compared to the Heap1 scenario, was faster by 20% when the number of robots $N_R = 10$, by 42% when $N_R = 25$ and by 56% when $N_R = 50$.

4.1.2 Local broadcasters

Similarly to the solitary swarms, the performance of local broadcasters was generally better when the number of worksites was larger (Figure 4.2). However, there were two types of condition that caused an exception to this rule: a lack of recruitment in small swarms and congestion in large swarms.

When the swarms were small ($N_R = 10$), the robots found all scenario types with the same D similarly difficult (see Figure 4.2, orange line). Because robots became more dispersed in the environment over time, the probability of recruitment decreased as an experimental run progressed. When the number of robots was small, it was often the case that one or two worksites were depleted by utilising recruitment at the beginning of an experimental run and the remaining worksites were discovered much later and processed in a more solitary fashion. In contrast, recruitment was more probable when swarms were larger, meaning that different sub-groups of the swarm exploited worksites in parallel, and therefore consumed them faster in the Heap4 and Scatter25 scenarios, compared to the Heap1 or Heap2 scenarios.

On the other hand, when the swarms were large ($N_R = 50$) but worksites were close together ($D \leq 9$m), recruitment occurred very frequently and caused congestion around worksites. Such *physical interference* between robots prevented them from accessing resources and more importantly, from finding other, non-congested worksites.

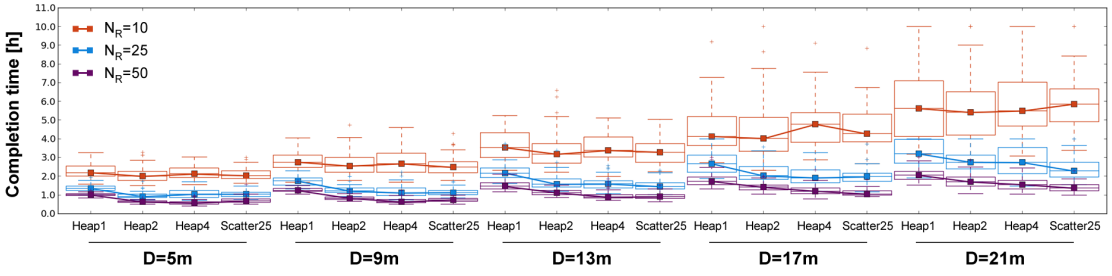


Figure 4.2: The static consumption task completion time of local broadcasters. The task completion time was shorter when worksites were numerous, as long the worksites were far away from the base and the number of robots $N_R \geq 25$.

4.1.3 Bee swarms

Contrary to the solitary swarms and local broadcasters, bee swarms generally found it harder to complete the consumption task when the number of worksites was large (Figure 4.3). This was mostly the case when the swarms were small ($N_R = 10$) or when worksites were far away from the base. Unlike robots from the other swarms, bee robots returned to the base to recruit when they discovered a worksite, meaning that they had to spend additional time in order to exploit the worksite. Moreover, bee swarms were likely to suffer from *exploitational interference*[1] between robots, where a worksite was depleted by other members of the swarm while the robot was recruiting in the base. Exploitational interference caused the recruiter and its recruits to travel to and search for a worksite that was no longer there, and thus to waste time that could have been spent for exploring the environment or for working. The interference was stronger in environments with more worksites, as there was a higher probability of them being discovered and depleted by other robots.

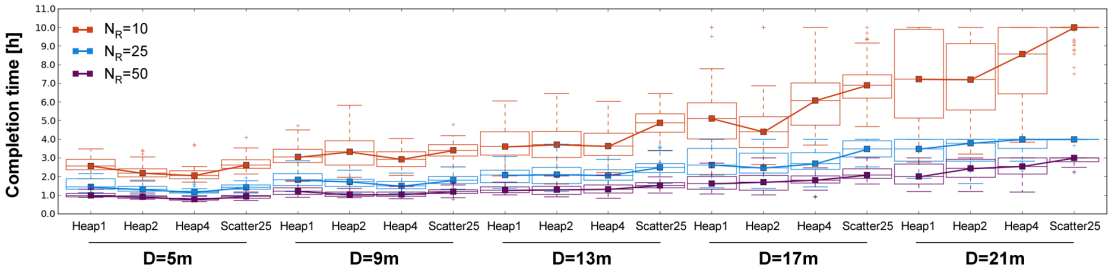


Figure 4.3: The static consumption task completion time of bee swarms. The task completion time was longer when worksites were numerous, especially when worksite distance was large or when swarms were small.

4.1.4 The performance of swarms relative to each other

The performance of the explored control strategies relative to each other was dependent on the type of the experimental environment (Figure 4.4). Solitary swarms completed the consumption task faster than the other swarms in the Scatter25 scenarios when worksites were close to the base. In these environments, recruitment, utilised by both local broadcasters and bee swarms, led to physical interference between robots, since the probability to discover worksites and to recruit to them was high. Additionally, bee swarms also experienced exploitational interference, where robots were recruiting to worksites that were meanwhile depleted by others. The disadvantage of local broadcasters and bee swarms was stronger when there were more robots in the swarm, i.e., when the interference between robots was stronger.

[1] Physical and exploitational interferences were first described in Pitonakova, L., Crowder R. & Bullock, S. (2014). "Understanding the role of recruitment in collective robot foraging", see Appendix H. The exploitational interference was named "environmental" in the paper.

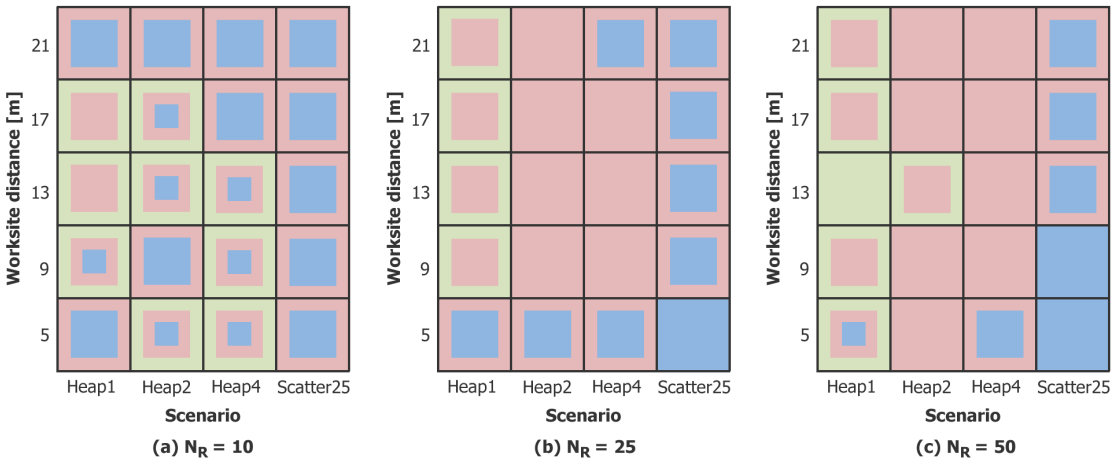


Figure 4.4: Winning strategies that completed the static consumption task the fastest in different environments using (a) 10, (b) 25 and (c) 50 robots. The strategies are identified by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix B, Figures B.1 - B.3.

On the other hand, Heap1 scenarios represented the most difficult environments, where only a single worksite existed and resource was thus difficult to find. Local broadcasters and bee swarms thus outperformed the solitary swarms in most Heap1 environments. Finally, in the intermediate Heap2 and Heap4 scenarios, bee swarms generally could not perform as well as the other swarms, both due to exploitational interference and due to the fact that they had to spend additional time traveling to the base to recruit. In these environments, solitary swarms and local broadcasters did similarly well when D was small, while more difficult environments with large D favoured local broadcasters.

It is also notable that the number of scenarios where multiple strategies did similarly well was higher when the swarms were small (compare Figure 4.4a and c). The probability of robots meeting each other was lower when there was a smaller number of robots, causing communication between local broadcasters and in the bee swarm to be less common. Therefore, the positive effects of recruitment, that could help the robots to exploit worksites that were hard to find, as well as the negative effects of interference, that could prevent robots from working effectively, were less pronounced when swarms were smaller.

Moreover, large swarm size affected the swarms differently in different environments. For both solitary swarms and local broadcasters, a larger number of robots usually improved performance more significantly in the Scatter25 compared to the Heap1 environments, while larger bee swarms enjoyed similar advantage across different scenarios, given the same D (Figure 4.5). For example, swarms of 50 solitary robots enjoyed a 55% to 62% reduction in task completion time compared to 10-robot swarms in the Heap1 scenario

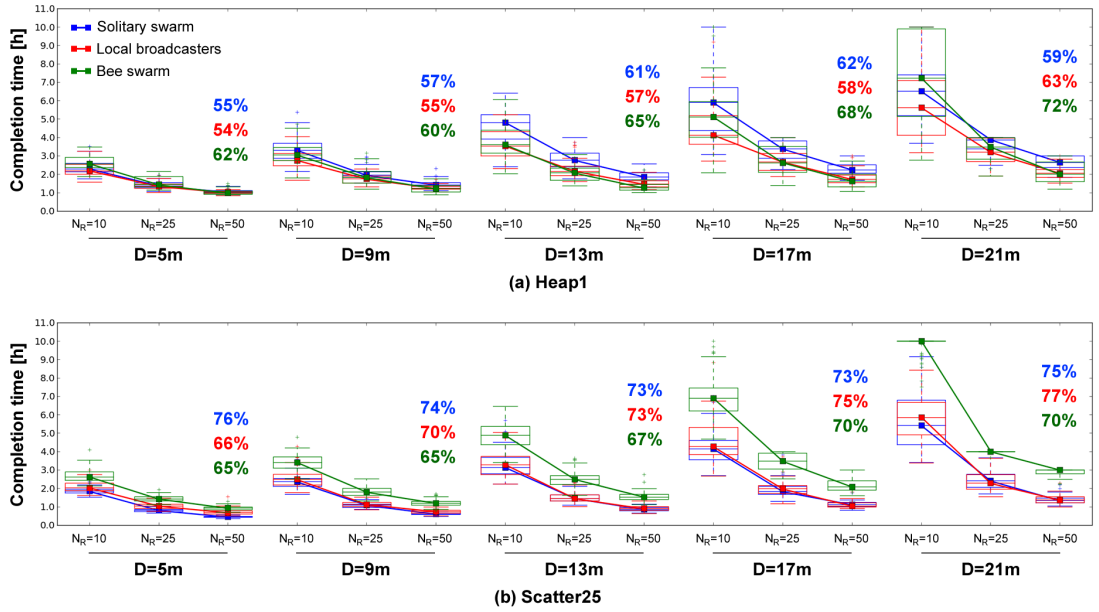


Figure 4.5: The effect of swarm size on the static consumption task completion time. The numbers above the box plots indicate how much the completion time was reduced by for each swarm when using 50 compared to 10 robots. All swarms completed the task faster when there were more robots used. Solitary swarms and local broadcasters benefited from a large swarm size more when there were more worksites in the environment. Bee swarms benefited from a large swarm size similarly regardless of the number of worksites.

and 73% to 76% reduction in the Scatter25 scenario. Similarly, swarms of 50 local broadcasters completed the task 54 to 63% faster than 10 robots in Heap1 scenario and 67% to 77% faster in Scatter25 scenario. However, 50-robot bee swarms completed the task around 62 to 72% faster than 10 robots in both Heap1 and Scatter25 scenarios. This result suggests that the performance of swarms that utilise recruitment in a designated location, i.e., the base, is less affected by the structure of the environment.

Robot swarms performing tasks similar to the consumption task explored here have been investigated by other authors. For example, robots had to find resource patches to "consume" (Balch and Arkin, 1994), they had to complete "jobs" on a manufacturing floor (Gerkey and Mataric, 2003; Dahl et al., 2009; Sarker and Dahl, 2011), or respond to discovered intruders (Raman, 2014). In line with the results presented here, it has been shown that communication between robots can improve performance when worksites are difficult to find (Balch and Arkin, 1994), and that swarm performance increases sublinearly with swarm size as a result of interference between robots (Dahl et al., 2009; Sarker and Dahl, 2011).

4.2 Information flow analysis

Analysing when and how information is acquired by a swarm and how it spreads between robots is the first step towards understanding why some control strategies are more suitable than others in a given environment. In this section, two characteristics of a swarm that relate to its information flow are introduced: *the scouting efficiency* and *the information gain rate*. Scouting efficiency refers to the ability of a swarm to discover new information in the environment. Information gain, ΔI, is related to both scouting efficiency and a recruitment strategy of a swarm and characterises the ability of robots to acquire and spread information. The rate of change in ΔI over time is described as the information gain rate, i.

4.2.1 Scouting efficiency

Each robot control strategy is associated with a certain scouting behaviour. All strategies explored here use Lévy movement (see Section 3.1.2) in order to search the environment. However, scouts in the bee swarm periodically return to the base in order to check whether there are robots recruiting to a worksite (see Section 3.1.4). This limits the amount of time they spend scouting [2].

A swarm's scouting efficiency can be approximated by measuring the time of the first worksite discovery in a given experimental run. The longer it takes a swarm to discover its first worksite, the worse scouting efficiency it has. The time of the *first*, rather than of the *last* or *average* worksite discovery is evaluated, in order to prevent interference between robots from affecting the measured ability of a swarm to scout.

While all swarms were less efficient at scouting in environments where worksites were far away from the base, the scouting efficiency of bee swarms was affected more significantly than that of other swarms (Figure 4.6). This trend was the strongest in Heap1 environments, where it was very difficult to discover a worksite, and it was weaker in other Heap environments (see Appendix B, Figures B.4 and B.5). The scouting efficiency of the bee swarms differed from that of other swarms more strongly when swarms were small, i.e., when $N_R = 10$ (compare Figure 4.6a and c). On the other hand, in Scatter25 environments, where worksites were numerous and thus easy to find, bee swarm's scouting efficiency was affected in a similar fashion than that of other swarms (Figure 4.7).

[2] The scouting time, T_S, of the bee swarm was set to best fit the environments explored here (see Appendix A). A more complex algorithm could use an adaptive value of T_S in order to deal with the problem of exploring large work arenas effectively. However, this would introduce more parameters to the algorithm and would not remove the fact that bee swarm scouts would have to return to the base, potentially missing the opportunity to scout more, while scouts of other swarms could continue exploring the environment.

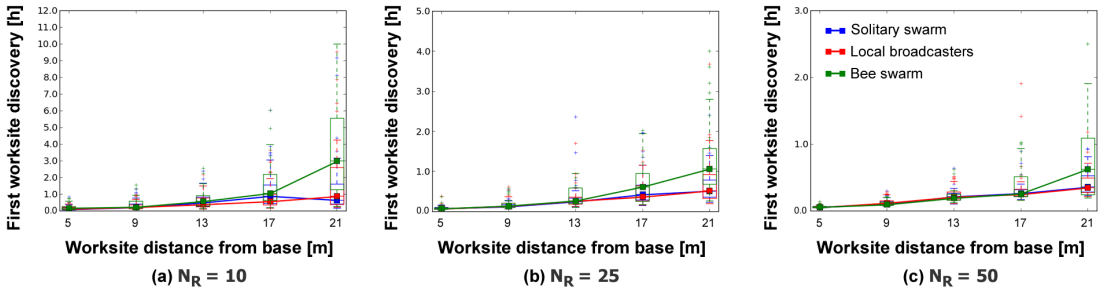


Figure 4.6: Time of the first worksite discovery in the static consumption task, Heap1 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other control strategies.

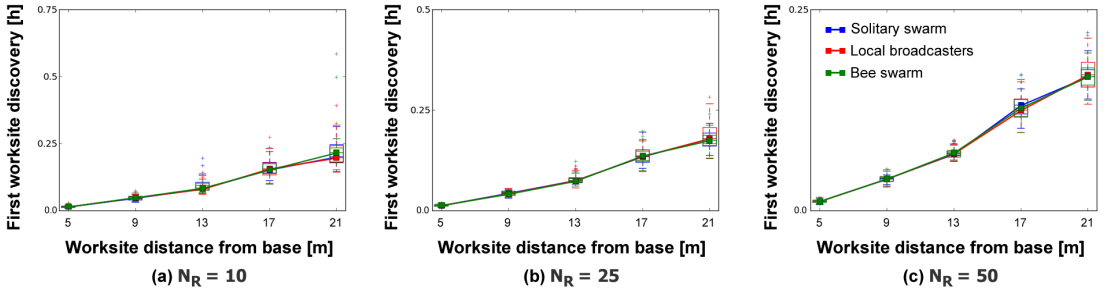


Figure 4.7: Time of the first worksite discovery in the static consumption task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was similar for all swarm types.

4.2.2 Information gain

The amount of information that the swarm has at a given point in time is defined as:

$$I(t) = \sum_{W}^{N_A} S_W(t) \tag{4.1}$$

where N_A is the number of active (i.e., not depleted) worksites in the environment and $S_W(t)$ is the number of robots that know about a worksite W at time t. Note that this definition of "information" differs from that traditionally used in engineering (Shannon, 1948), where information is measured in the context of transmission of symbols drawn from a finite pool of possibilities (Hartley, 1928). Since a piece of data about a swarm's environment can have an infinite number of possible values, for example, real-value coordinates that represent a worksite location, information here is represented in terms of the number of robots that have acquired a piece of data, i.e., the number of data pieces that the swarm possesses.

The information gain[3], ΔI, of a swarm represents the change in the amount of information that a swarm has. It is defined as:

$$\begin{aligned} \Delta I(t) &= I(t) - I(t-1) \\ &= \sum_{W}^{N_A} [S_W(t) - S_W(t-1)] \end{aligned} \tag{4.2}$$

We can obtain a normalised information gain, $\Delta I(t)'$, by dividing ΔI by the number of robots, N_R:

$$\Delta I(t)' = \frac{\Delta I(t)}{N_R} \tag{4.3}$$

By measuring information gain, we can identify when scouts find new worksites or when robots are recruited for work. In these cases, a swarm gains new information and ΔI is positive. Similarly, when robots abandon active worksites, and are assumed to no longer "know" about them, they have a negative information gain. When no information is gained or lost, $\Delta I = 0$.

Other information metrics for multi-agent systems exist, such as *transfer entropy* (TE) and *local information storage* (LIS) (Wang et al., 2012; Miller et al., 2014). Inspired by Shannon's understanding of information (Shannon, 1948), TE and LIS measure how communication affects the state of an agent based on the states of other agents that it receives information from. They can thus predict the next state of agents in tightly coupled multi-agents systems, such as a group of robots performing flocking. The "state" of a robot can for example represent its current discretised velocity or orientation. Discretisation is necessary, since representation of information in Shannon's sense requires a finite number of possible values that a variable can take (Hartley, 1928; Shannon, 1948). There are a few problems with this approach when it comes to measuring information flow in swarms. Firstly, because it measures the coupling between "states" of two agents at two different time steps, transfer entropy is to some extent a proxy measure of information flow. On the other hand, information gain, ΔI, is calculated as a change in the number of informed robots and it thus directly captures the amount of knowledge that the swarm gains or looses at a given point in time. Secondly, since ΔI is based on the number of informed robots, it does not require us to define what a robot "state" is in relation to having information. Informed robots may be performing a number of different operations, for example, gathering resources, traveling to the

[3]Information gain is a more general metric inspired by *information value* that has been used in Pitonakova, L., Crowder R. & Bullock, S. (2016). "Information flow principles for plasticity in foraging robot swarms" (see Appendix I) to analyse information flow in swarms that were choosing which worksites to forage from based on worksite utilities.

base, etc. Thirdly, as it will be demonstrated below, information gain can be directly related to the amount of uncertainty cost that the swarm pays (see Section 4.3.2, Equations 4.8 and 4.9) and thus to the ability of the swarm to turn information into reward (see Section 4.3.4, Equation 4.18). This allows us to not only characterise information flow, but also to relate it to swarm performance. To the best knowledge of the author, no study has demonstrated so far how and whether entropy can be directly related to swarm performance in missions where the resulting collective behaviour is dependent on factors other than the ability of robots to observe each other's actions and coordinate their behaviour.

The information gain time series looks very different for different swarms and environments. In Scatter25, where it is relatively easy to discover worksites as they are numerous, all swarms generate a large information gain, especially at the beginning of experimental runs, when scouting is the most successful (Figure 4.8). Solitary robots maintain $\Delta I' > 0$ until all worksites are depleted, as the robots are rather evenly spread across the work arena and usually do not get in each other's way while exploring and exploiting the environment. The region of the graph during which $\Delta I' > 0$ is referred to as a *positive information gain region*. Solitary swarms have a single positive information

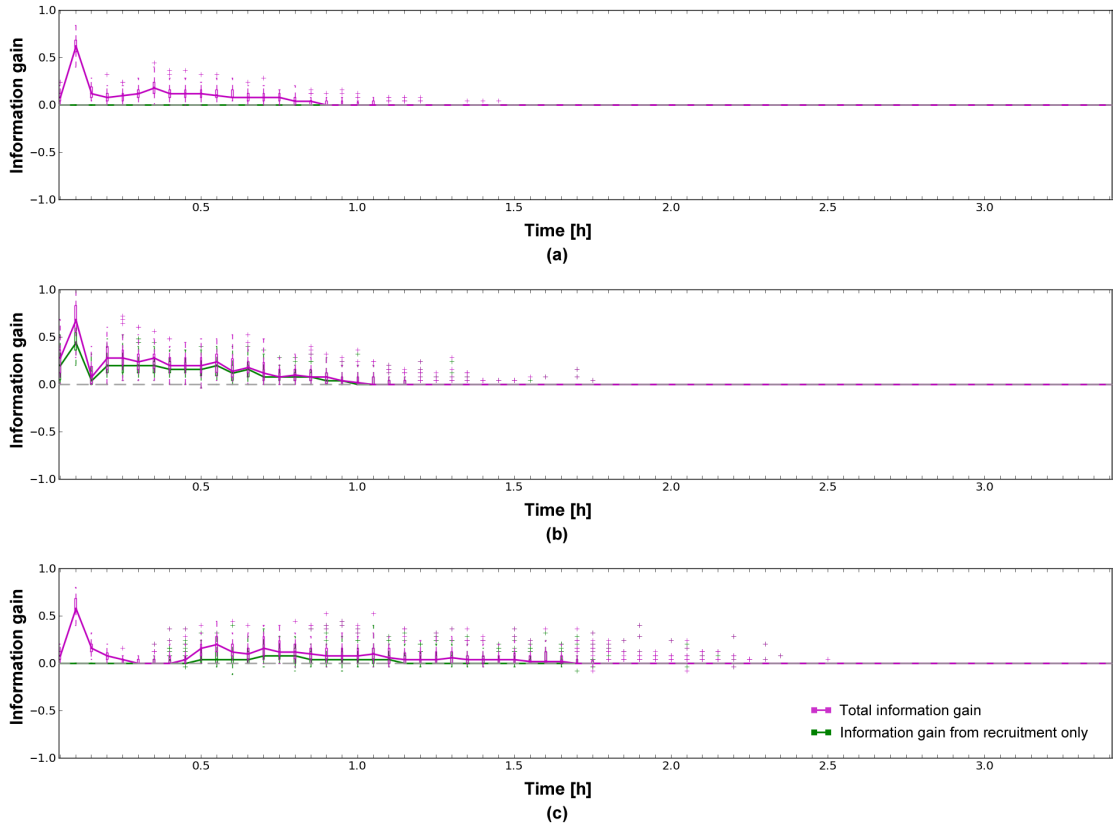


Figure 4.8: Normalised information gain of (a) 25 solitary robots, (b) 25 local broadcasters, (c) 25 bee robots in the static consumption task, Scatter25 scenario with $D = 9$m.

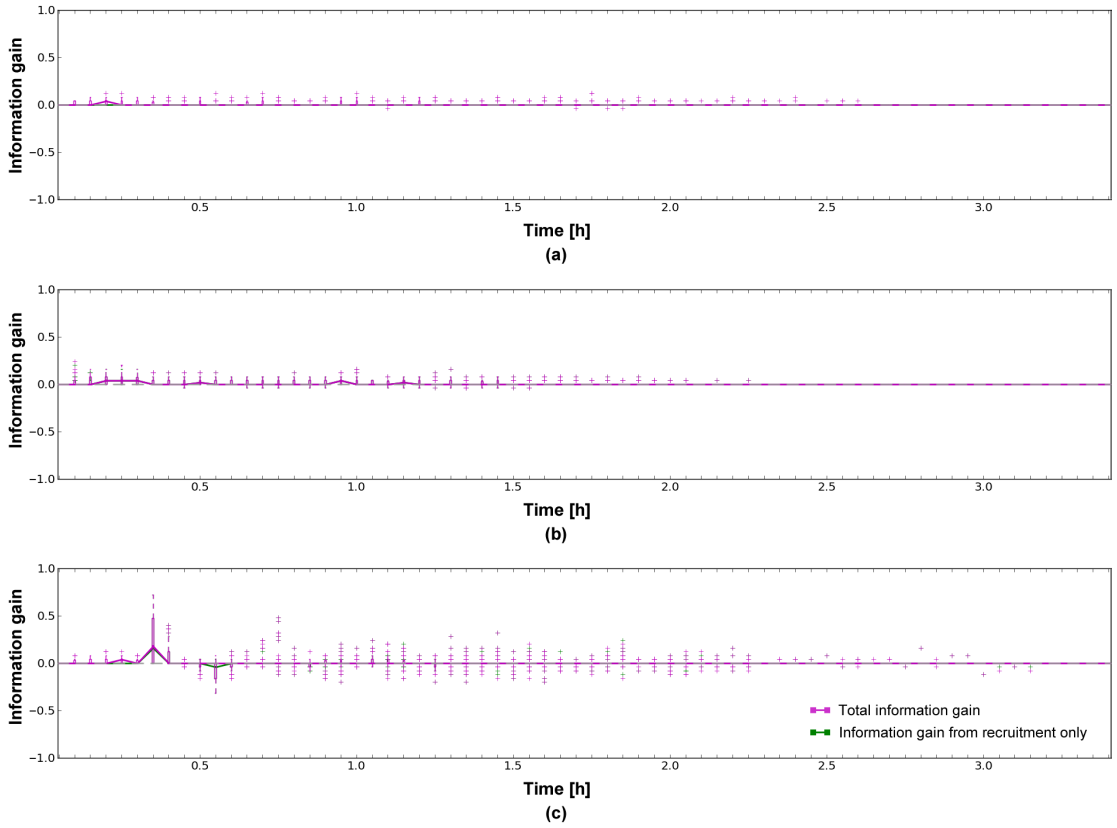


Figure 4.9: Normalised information gain of (a) 25 solitary robots, (b) 25 local broadcasters, (c) 25 bee robots in the static consumption task, Heap1 scenario with $D = 9$m.

gain region. Local broadcasters, on the other hand, start recruitment immediately after initial worksite discoveries are made, which leads to physical and exploitational interference between robots and a significant decrease in information gain after a few minutes, while the robots are avoiding each other and searching for depleted worksites that they were recruited to. There are two positive information gain regions, the larger one appearing at the beginning of the simulations. Finally, bee robots learn about worksites in a solitary fashion at the beginning of each run, and recruit each other in the base later. Similarly to local broadcasters, bee swarms show two positive information gain regions in this scenario, but the regions are further apart than those of local broadcasters, since bee swarm robots need additional time in order to travel to the base and recruit.

In the Heap1 scenario, where there is only a single worksite to discover, $\Delta I'$ of all swarms is significantly smaller than that in the Scatter25 scenario. Solitary swarms find it the most difficult to obtain information about the worksite (Figure 4.9a). There are many $\Delta I'$ outliers that correspond to isolated events when robots find the worksite, but the only positive region, that indicates a clear trend of worksite discovery, appears at the beginning of the experimental runs and it is very small. Local broadcasters show an improved ability to discover the worksite, with a number of small positive $\Delta I'$ regions

spread across the simulation run (Figure 4.9b). The time series also indicates that most of the information gain of local broadcasters is due to recruitment. A small number of robots usually discover the worksite at the beginning of an experimental run and broadcast it to other robots nearby. Finally, bee swarms show the strongest ability to gain information in this scenario (Figure 4.9c). A small $\Delta I'$ region at the beginning of the time series represents an initial worksite discovery by scouts, followed by a larger positive region, representing recruitment of more workers in the base. A small negative region appears after some time, indicating that some recruits abandon the worksite as a result of their inability to access it due to congestion.

4.2.3 Information gain rate

It is apparent that the investigated control strategies differ in how quickly their positive information gain regions can grow, especially in environments such as Heap1, where worksites are difficult to discover. To characterise this growth, *information gain rate*, i, is calculated based on the total information gain $\Delta I'$ of positive regions and the length of these regions.

In order to calculate i, the information gain time series is first down-sampled into time intervals T_i seconds long by summing the information gain of all robots in each interval. Down-sampling $\Delta I'$ in this way makes it possible to identify and measure trends in $\Delta I'$, since individual information gain events, such as a robot finding a worksite or a robot being recruited, that usually occur a few seconds apart, are grouped together into discrete time intervals. Positive regions are then distinguished from the rest of the down-sampled time series by considering intervals during which the down-sampled information gain, ΔI^*, remains positive. Information gain rate i_P in each positive region is defined as:

$$i_P = \frac{\sum_{T=0}^{T_P} \Delta I^*(T)}{T_P} \tag{4.4}$$

where T_P is the length of a positive region in seconds and $\Delta I(T)^*$ is the total, down-sampled, information gain in a given time interval.

The information gain rate of a swarm i is the maximum value of i_P measured in an experimental run:

$$i = \max\left(i_P\right) \tag{4.5}$$

The *maximum*, rather than *median* or *average* i_P is used, because it is less likely to be affected by physical and exploitational interference that result from recruitment.

The time interval T_i, used for down-sampling the information gain time series, is a parameter to the information gain rate calculation, set to $T_i = 60$s. Appendix C shows that while this particular value makes the distinction between information gain rate of various swarms the strongest, using a different value does not affect the order of the swarms based on i.

In solitary swarms, worksite discoveries are more probable when worksites are abundant or when the work arena is small. Consequently, the information gain rate of solitary swarms increases with the number of worksites, N_W, and decreases with worksite distance, D (Figure 4.10, blue line). A similar trend can be observed for local broadcasters (Figure 4.10, red line). On the other hand, the information gain rate of bee swarms (Figure 4.10, green line) varies less across scenarios and generally does not increase with N_W, unless the worksites are far away from the base ($D = 21$m) or when swarms are small ($N_R = 10$, see Appendix B, Figure B.6). Since bee swarms use a designated location to spread information, they are able to achieve a relatively high information gain rate in difficult environments like Heap1, where the probability of a robot discovering a worksite is low. However, their information gain rate usually cannot increase further in easier environments, such as Scatter25, due to interference between robots that prevents them from working and thus from recruiting.

Consequently, the information gain rate of bee swarms is usually the highest in the Heap environments, followed by local broadcasters and solitary swarms (see Figure 4.10 and Appendix B, Figures B.6 and B.7). On the contrary, local broadcasters reach the highest information gain rate in the Scatter25 environments, as they recruit near worksites and the probability of such recruitment is high when worksites are numerous, especially when they are also close together ($D <= 9$m). As D increases, information becomes more difficult to obtain and spread and the information gain rate of all three swarms becomes more similar.

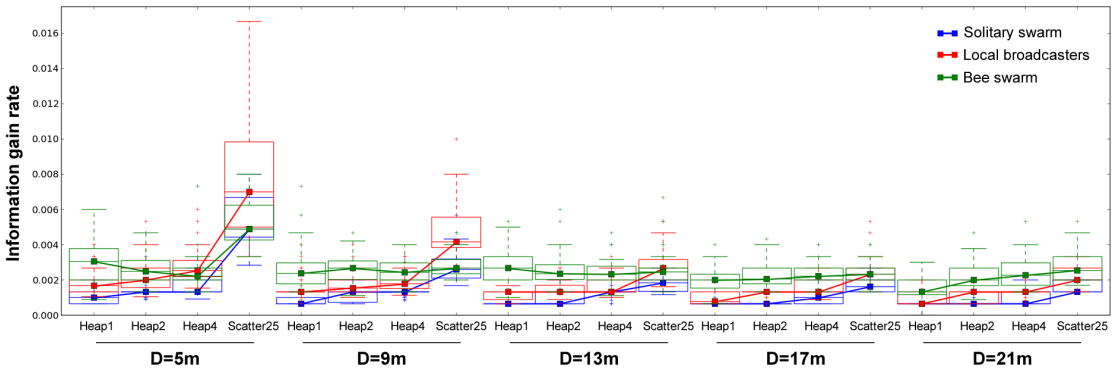


Figure 4.10: Information gain rate, i, of 25-robot swarms in the static consumption task. Bee swarms enjoyed the highest i in most environments, apart from the Scatter25 scenarios when $D \leq 9$m.

4.2.4 Information flow and swarm performance

It is important to point out that the ability of robots to find and spread information quickly does not always result in the ability of the swarm as a whole to perform a task efficiently. There are notable differences between which swarm achieves the highest information gain rate and which swarm completes the task the quickest. For example, even though bee swarms achieve the highest i in the Heap environments, they do not outperform local broadcasters in any of the explored environments. On the other hand, solitary swarms, that achieve the lowest information gain rate in all environments, are able to complete the consumption task faster than any other swarm when worksites are easy to find.

In order to better understand the discrepancy between information flow and swarm performance, it is important to investigate what exactly happens to information once it reaches robots and how it is utilised by them. The remained of this chapter is devoted to this investigation.

4.3 Cost analysis

It is apparent from the comparison between information flow and task completion time of swarms that there exists a certain pressure from the environment that prevents the robots from converting the information that they have about worksites into a reward from those worksites. Moreover, this pressure acts differently on swarms with different control strategies. It can be said that different swarms pay different *costs* for utilising information. In this chapter, the work cycle of a robot is first described, and the costs paid at each stage of the work cycle are defined. It is then demonstrated that the costs, when added up together, explain the difference between the reward that a swarm is receiving at a given point in time and the *expected* reward that a swarm *should* receive based on the information that it carries. Note that these costs are calculated by a god-like observer in an effort to evaluate the system - they are not intended to be calculated on the fly by individual robots. Together with the nature of its information flow, tendency of a control strategy to incur costs define a set of swarm characteristics that determine whether a control strategy is suitable for a given task and environment. The interplay between environmental and swarm characteristics is explained in Section 4.4.

4.3.1 The robot work cycle

In both the consumption and the collection task, a robot needs to find information about a worksite and utilise that information in order to obtain reward. In the consumption task, reward can be obtained directly from the worksite, while a robot remains close to it. In the collection task, a robot obtains reward when it drops off resource in the base.

A robot's work cycle (see Figure 4.11) can be generalised for both task types as follows. A robot starts by being *unemployed* (U) and searches the environment for information. When it discovers a worksite, either by itself or as a result of being recruited, it *subscribes* (S) to that worksite. It then travels to it and becomes *laden* (L) with resource. It starts *earning* (E) reward when it reaches a *reward generator*. In the *consumption* task, the reward generator is the worksite itself and laden robots immediately become earning robots. In the *collection* task, the reward generator is the base and laden robots need to travel there in order to earn reward. Note that the total number of robots in a swarm, $N_R = U + S$ and that $S \geq L \geq E$.

During each stage of the work cycle, a robot has the potential to incur certain costs (see Figure 4.11). There are three types of cost: *uncertainty cost*, C_U, (Section 4.3.2) incurred by unemployed robots that do not know where work is located, *misplacement cost*, C_M, (Section 4.3.2) that all subscribed robots pay until they reach a reward generator and start earning reward, and *opportunity cost*, C_O, (Section 4.3.3) incurred by robots that are subscribed to depleted worksites and are thus unable to find or perform work.

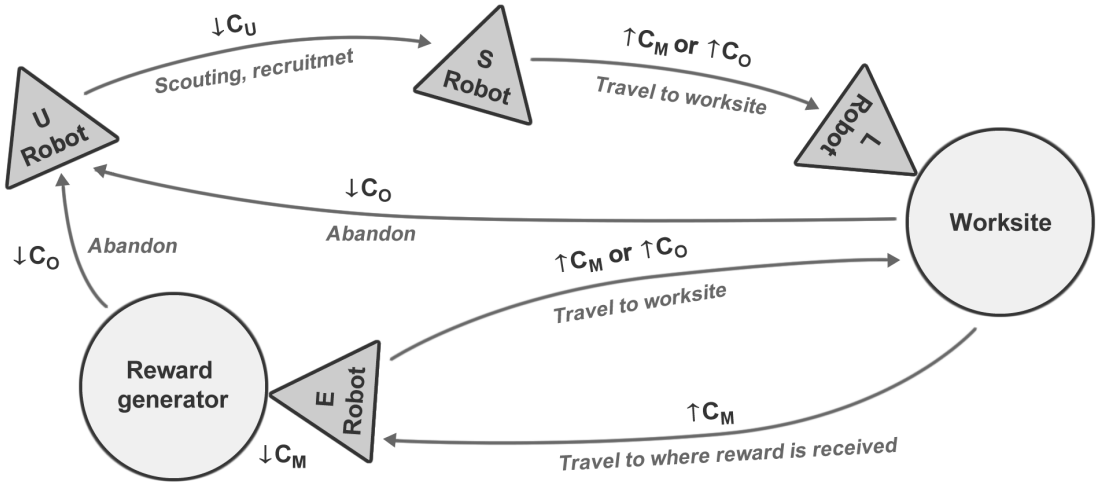


Figure 4.11: The robot work cycle, along with the costs incurred at each stage. The robot starts by being unemployed (U). It becomes subscribed (S) once it learns about a worksite, decreasing the uncertainty cost, C_U, of the swarm. It then travels to the worksite while incurring misplacement cost, C_M, after which it becomes laden (L) with resource. When close to a reward generator, the robot earns (E) reward and decreases the swarm's misplacement cost. During the consumption task, the reward generator is the worksite itself and a robot thus becomes laden and starts earning reward at the same time, incurring no further C_M. During a collection task, a robot contributes to the total, C_M, of the swarm until it reaches the base. Additionally, a robot may incur opportunity cost, C_O, if it is traveling to a worksite that has been depleted. When the robot abandons the depleted worksite and becomes unemployed again, the total C_O of the swarm decreases.

Figure 4.12 shows an example of how these costs are incurred by robots that utilise recruitment. At the beginning of a run, all robots are unemployed, paying the maximum amount of uncertainty cost. C_U decreases when robots learn about a worksite, while C_M increases as some of those robots are recruits that are not yet located at the worksite. When one worksite gets depleted (just before the first hour in Figure 4.12), the total uncertainty cost decreases, since there is one less active worksite that the swarm needs to know about. However, robots still subscribed to the depleted worksite incur an opportunity cost, until they determine that the worksite is in fact depleted and they abandon it. The task is completed when all worksites are depleted (at around 1.5 hours in Figure 4.12), rendering all costs to be 0.

Quantifying these costs first requires calculating the amount of reward, r, available per worksite and per robot:

$$r = \frac{R_T}{N_R \times N'_W} \tag{4.6}$$

where R_T is the total amount of reward available in the environment, N_R is the total number of robots and N'_W is the number of worksites at the beginning of an experiment. During experiments with static environments, $R_T = 100$ (see Section 3.1.1). If reward r could be obtained by all robots from all worksites at the same time, the task would be completed.

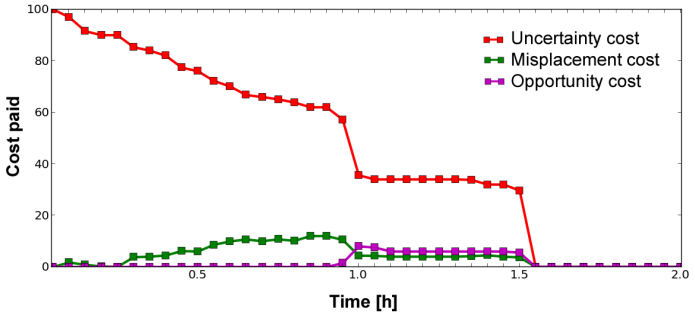


Figure 4.12: An example of the costs incurred by robots over time, using a single experimental run with 25 local broadcasters in the static consumption task, Heap2 scenario, $D = 9$m. The uncertainty cost decreases as robots discover the two worksites. Recruitment causes some robots to be misplaced from a worksite that they know about and to thus incur misplacement cost. When the first worksite gets depleted (at around $t = 1$h), robots that are away from it pay opportunity instead of misplacement cost. Opportunity cost is paid until the robots are able to confirm that the worksite is not active anymore. All costs are equal to 0 when the second worksite gets depleted and the task is completed.

4.3.2 Uncertainty and misplacement costs

At the beginning of a run, all robots are unemployed and the swarm has no information about where reward is located. The swarm is therefore paying *uncertainty cost* C_U, equal to the total reward from all worksites (all worksites are active at this point). When a robot finds out about a worksite, the swarm's C_U decreases by r. At any given time, the amount of uncertainty cost a swarm pays is thus:

$$\begin{aligned} C_U &= \sum_{W=1}^{N_A} (N_R \times r) - \sum_{W=1}^{N_A} (S_W \times r) \\ &= \sum_{W=1}^{N_A} [(N_R - S_W) \times r] \end{aligned} \tag{4.7}$$

where N_A is the number of active worksites and S_W is the number of robots subscribed to a worksite W.

The change in uncertainty cost, ΔC_U, relates to the swarm's information gain, ΔI (Section 4.2.2, Equation 4.2), in the following way:

$$\begin{aligned} \Delta C_U &= C_U(t) - C_U(t-1) \\ &= [\sum_{W=1}^{N_A(t)} (N_R \times r) - \sum_{W=1}^{N_A(t)} (S_W(t) \times r)] \\ &\quad - [\sum_{W=1}^{N_A(t-1)} (N_R \times r) - \sum_{W=1}^{N_A(t-1)} (S_W(t-1) \times r)] \\ &= [\sum_{W=1}^{N_A(t)} (N_R \times r) - \sum_{W=1}^{N_A(t-1)} (N_R \times r)] - \Delta I \times r \end{aligned} \tag{4.8}$$

Or:

$$\Delta I \times r = -\Delta C_U + [\sum_{W=1}^{N_A(t)} (N_R \times r) - \sum_{W=1}^{N_A(t-1)} (N_R \times r)] \tag{4.9}$$

In other words, the swarm's information gain between time steps t and $(t-1)$ is directly proportional to the sum of the decrease in the swarm's uncertainty cost and the change in the total available reward from all active worksites. If the number of active worksites at time step t remains the same as in time step $(t-1)$, i.e., when no worksites are depleted or added to the environment, then $\sum_{W=1}^{N_A(t)}(N_R \times r) - \sum_{W=1}^{N_A(t-1)}(N_R \times r) = 0$ and $\Delta I \times r = -\Delta C_U$.

As was mentioned in the robot work cycle description above, a robot might need to travel to a worksite or to a reward generator in order to obtain reward. During the time period between when the robot subscribes to a worksite and when it starts earning reward from it, the robot is misplaced from the reward generator and pays a *misplacement cost*, C_M:

$$C_M = \sum_{W=1}^{N_A} [(S_W - E_W) \times r] + \sum_{W=1}^{N_D} [(L_W - E_W) \times r] \tag{4.10}$$

where E_W is the number of robots *earning* reward from a worksite W, N_D is the number of depleted worksites and L_W is the number of robots laden with resource from a worksite W. The first term on the right hand side of Equation 4.10 represents the misplacement cost that subscribed robots pay from active worksites, either during traveling to worksites, or, in the case of the collection task, during traveling to the base to unload resource (see Figure 4.11). The second term represents cases when robots laden with resource from depleted worksites travel to the base during the collection task [4].

The relationship between a reduction in uncertainty cost and an increase in misplacement cost determines what portion of robots that learn about a worksite are able to obtain reward from it. In an ideal case, no misplacement cost is paid and reduction in uncertainty immediately leads to an increase in reward. We can characterise this relationship in terms of the *misplacement cost coefficient*, m, as:

$$m = \frac{C_M}{\sum_{W=1}^{N_A}(S_W \times r)} \tag{4.11}$$

When $m = 0$, a decrease in uncertainty cost is fully turned into reward, i.e., $C_M = 0$. When $m = 1$, all robots that know about worksites are misplaced from a reward generator and no reward is obtained, i.e., $C_M = \sum_{W=1}^{N_A}(S_W \times r)$. Intermediate values of $0 < m < 1$ indicate that some robots are misplaced and some are receiving reward. Misplacement cost coefficient is affected both by the way in which a robot control strategy utilises information and by where in the work arena information is shared.

Solitary robots do not pay C_M during the consumption task, since they do not recruit and since scouts are already present at a worksite when they learn about it. In swarms that do communicate, C_M is incurred by recruited robots until they reach a worksite advertised to them. Additionally, in the bee swarms, scouts incur C_M, as they travel to the base and back in order to recruit.

In 25-robot swarms, the misplacement cost coefficient m is generally the highest in bee swarms and it is always 0 in solitary swarms (Figure 4.13). However, when the number

[4] The number of laden, rather than subscribed robots is used in the second right hand side term of Equation 4.10, as a robot that depletes a worksite during the collection task abandons it immediately (i.e., it is not subscribed to it anymore), but it still needs to travel to the base in order to obtain reward.

of worksites is small, most notably the Heap1 scenarios, or when worksites are very close to the base (D = 5m in all Heap scenarios), local broadcasters experience congestion near worksites and their m is higher or similar as that of bee swarms. The congestion that local broadcasters experience is caused by the fact that a local broadcaster worker recruits the whole time while it is depleting a worksite. In environments where worksites have large volumes and are close to each other, such a worker has a high probability of recruiting too many robots, preventing some of them from accessing the worksite and causing them to incur misplacement cost for a long period of time. On the other hand, bee robots recruit in the base and therefore access worksites in a less congested fashion.

A similar comparison between m of the different swarms can be made when 10 and 50 robots are used (see Appendix B, Figures B.8 and B.9).

4.3.3 Opportunity cost

On some occasions, an unladen robot becomes subscribed to a worksite that has been depleted. This can happen either to new recruits, or to any robots during the collection task while they are traveling from the base back to a depleted worksite. During this time, a robot is missing an opportunity to explore the environment, and it is thus incurring an *opportunity cost*, C_O:

$$C_O = \sum_{W=1}^{N_D} [(S_W - L_W) \times r] \tag{4.12}$$

Note that robots that are laden with resource from a depleted worksite pay misplacement cost instead (see the second right hand side term of Equation 4.10).

Opportunity cost measures the negative impact of exploitational interference, i.e., the tendency of a swarm to commit to worksites that become depleted while robots are away

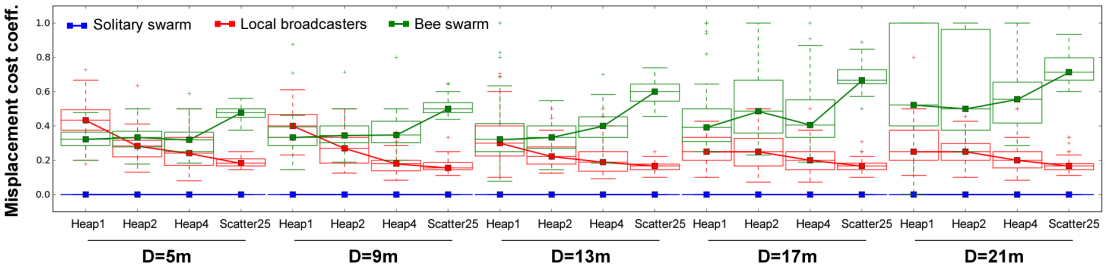


Figure 4.13: Misplacement cost coefficient, m, of 25-robot swarms in the static consumption task. Bee swarms usually had the highest m, apart from environments where the number of worksites and worksite distance were small. The m of solitary swarms was always 0.

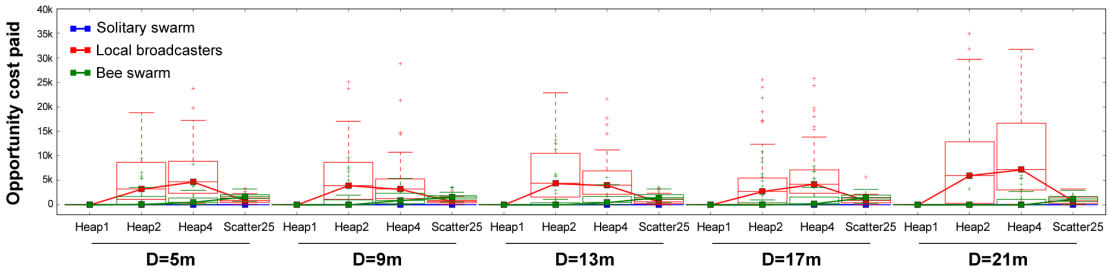


Figure 4.14: Opportunity cost, C_O, of 25-robot swarms in the static consumption task. Local broadcasters paid a larger C_O than the other swarms in Heap environments, while in Scatter25 environments, C_O of bee swarms was the largest. Solitary swarms did not pay C_O.

from them. The more a swarm overcommits to a worksite by having too many robots subscribed to it, the higher opportunity cost it can potentially pay when that worksite becomes depleted. The cost is related to the number of robots that a single piece of discovered information can affect, i.e., to information gain rate, as well as to how robots recruit.

When comparing the opportunity cost incurred by different swarms, it is important to only consider the cost while there are active worksites available in the environment. When the last worksite is depleted, robots that are still subscribed to it are not missing an opportunity to do more work, since the environment is empty. Consequently, when C_O of different swarms is compared, it is always shown as 0 in Heap1 scenarios. Figure 4.14 depicts C_O paid by 25-robot swarms. Results for swarms of 10 and 50 robots are shown in Appendix B, Figures B.10 and B.11.

Solitary robots do not incur C_O during the consumption task because they do not recruit, meaning that when a worksite is depleted, all robots that are subscribed to it immediately become aware of that fact and abandon the worksite. On the other hand, local broadcasters, that recruit near worksites continuously until it gets depleted, pay the highest C_O in the Heap environments. Their C_O is usually higher in the Heap4 environments when D is large and the number of robots is small ($N_R = 10$), i.e., when recruitment is less probable and when a smaller number of robots usually process a single worksite. Because worksite processing takes longer in these cases and because there are more potential recruits exploring the environment, there is a higher chance that a robot will be recruited to a worksite that is about to be depleted. On the other hand, in Scatter25, worksites have small volumes and are thus depleted quickly, which decreases the probability of new workers being recruited to a worksite that is almost empty. Finally, bee swarm robots incur C_O that is similar across the different environments. Since they recruit in the base, robot arrival times to worksites are more similar to each other, compared to when local broadcasters recruit.

4.3.4 Costs and reward

In this section, the relationship between how the information that a swarm has, the costs that it incurs and the reward that it obtains, is formalised. First, the amount of *actual* reward ΔR, that a swarm is earning at a given point in time needs to be calculated:

$$\Delta R = \sum_{W=1}^{N_W} (\rho \times E_W) \tag{4.13}$$

where $N_W = N_A + N_D$ is the total number of worksites (active and depleted) and ρ is the reward intake rate, i.e., the amount of reward gained per second by an earning robot E. In the consumption task, $\rho = 1/400$s. In the collection task, $\rho = 1$s (see the beginning of Chapter 3).

If robots could immediately turn information into reward, i.e., if they did not have to travel to worksites and did not suffer from contention for the same resource, the swarm could earn *expected reward* R':

$$R' = \sum_{W=1}^{N_W} (S_W \times r) \tag{4.14}$$

Equation 4.15 shows that the sum of the misplacement and opportunity costs is equal to the difference between the expected reward and the actual reward, multiplied by $\frac{r}{\rho}$. The term $\frac{r}{\rho} \times \Delta R$ refers to the actual per-robot reward that the swarm will receive in $1/\rho$ seconds. The equation signifies the fact that a swarm cannot utilise information about worksites for free - it has to pay costs associated with its control strategy and with the structure of the environment.

$$\begin{aligned}
C_M + C_O &= [\sum_{W=1}^{N_A} (S_W \times r) - \sum_{W=1}^{N_A} (E_W \times r) + \sum_{W=1}^{N_D} (L_W \times r) - \sum_{W=1}^{N_D} (E_W \times r)] \\
&\quad + [\sum_{W=1}^{N_D} (S_W \times r) - \sum_{W=1}^{N_D} (L_W \times r)] \\
&= [\sum_{W=1}^{N_A} (S_W \times r) + \sum_{W=1}^{N_D} (S_W \times r)] - [\sum_{W=1}^{N_A} (E_W \times r) + \sum_{W=1}^{N_D} (E_W \times r)] \\
&= \sum_{W=1}^{N_W} (S_W \times r) - \sum_{W=1}^{N_W} (E_W \times r) \\
&= R' - \frac{r}{\rho} \times \Delta R
\end{aligned} \tag{4.15}$$

The *potential reward*, R^*, is defined as a sum of the expected reward and a reward that could be received by all unemployed (i.e., non-subscribed) robots from all active worksites, if the unemployed robots knew where the worksites were located:

$$R^* = \sum_{W=1}^{N_W} [S_W \times r] + \sum_{W=1}^{N_A} [(N_R - S_W) \times r] \tag{4.16}$$

Since $N_W = N_A + N_D$, the potential reward can also be expressed as:

$$\begin{aligned} R^* &= [\sum_{W=1}^{N_A} (S_W \times r]) + \sum_{W=1}^{N_D} (S_W \times r)] + [\sum_{W=1}^{N_A} (N_R \times r) - \sum_{W=1}^{N_A} (S_W \times r)] \\ &= \sum_{W=1}^{N_A} (N_R \times r) + \sum_{W=1}^{N_D} (S_W \times r]) \end{aligned} \tag{4.17}$$

The sum of all three costs, uncertainty, misplacement and opportunity, is equal to the difference between the potential reward and the actual reward, multiplied by $\frac{r}{\rho}$ (Equation 4.18). Equation 4.18 formalises the relationship between the swarm's information flow, which affects the amount of uncertainty cost paid, the swarm's tendency to incur the misplacement and opportunity costs while utilising the information, and the reward that the swarm is able to extract from the environment at a given time.

$$\begin{aligned} C_U + C_M + C_O &= [\sum_{W=1}^{N_A} (N_R \times r) - \sum_{W=1}^{N_A} (S_W \times r)] \\ &\quad + [\sum_{W=1}^{N_A} (S_W \times r) - \sum_{W=1}^{N_A} (E_W \times r) + \sum_{W=1}^{N_D} (L_W \times r) - \sum_{W=1}^{N_D} (E_W \times r)] \\ &\quad + [\sum_{W=1}^{N_D} (S_W \times r) - \sum_{W=1}^{N_D} (L_W \times r)] \\ &= [\sum_{W=1}^{N_A} (N_R \times r) + \sum_{W=1}^{N_D} (S_W \times r)] - [\sum_{W=1}^{N_A} (E_W \times r) + \sum_{W=1}^{N_D} (E_W \times r)] \\ &= R^* - \frac{r}{\rho} \times \Delta R \end{aligned} \tag{4.18}$$

In an ideal world, where robots could locate themselves at worksites immediately upon learning about them, and where there would be no physical or exploitational interference between robots, the misplacement and opportunity cost would not exist. However, because swarms operate in the real world, where time and space do play a role, we need

to consider these costs in order to account for the discrepancy between the information a swarm possesses and the reward that it can receive at a given point in time. Furthermore, since robots do not know where worksites are located and they need to explore the environment, the way in which they decrease their uncertainty is also important. The Information-Cost-Reward framework, introduced in the next section, explains in more detail how the information flow and the tendency of robots to incur costs fit together and how they affect the performance of a swarm with a particular robot control strategy in a given environment.

4.4 The swarm-environment fit

Swarms employing different control strategies perform differently in each environment (see Section 4.1). The control strategies also differ in the way in which information is acquired by and shared between robots and in the costs that the robots incur when turning information into reward. There are four characteristics of a robot control strategy that affect the swarm's performance: its scouting efficiency, its information gain rate, and its tendency to incur misplacement and opportunity costs. While the scouting efficiency and information gain rate affect how uncertainty of a swarm about its environment decreases, the misplacement and opportunity costs characterise how effective a swarm is in turning information into reward. In this section, the relationship between these characteristics and the swarm's performance is explained.

4.4.1 Scouting efficiency

Scouting efficiency is related to the ability of a swarm to discover new information about worksites in the environment. Scouting efficiency of all three swarms is smaller when there are few worksites in the environment or when worksites are far away from the base (see Section 4.2.1). While solitary swarms and local broadcasters are affected by worksite distance in a similar manner, bee swarms suffer a much higher decrease in performance when worksite distance is large. The poor scouting efficiency of bee swarms is caused by the fact that bee swarm scouts periodically return to the base to check whether they could be recruited by informed robots. As a result, bee swarms perform poorly compared to the other swarms when the number of robots is small or when worksites are far away from the base (see Figure 4.4).

4.4.2 Information gain rate

Information gain rate, i, characterises how quickly a swarm is able to acquire new information, as well as how quickly the information can spread through the swarm. It is

related to the swarm's ability to scout, but also to its recruitment strategy. Because they use a designated area to exchange information, bee swarms enjoy the highest information gain rate in most environments, followed by local broadcasters and solitary robots (see Section 4.2.3). Additionally, the i value of bee swarms is more stable across different environments compared to other swarms.

A high information gain rate is advantageous in environments where it is difficult to discover worksites, most significantly in the Heap1 scenarios or when worksites are far away from the base. In these environments, local broadcasters and bee swarms achieve the best performance (see Figure 4.4). On the other hand, in environments where worksite discoveries are more probable, fast information gain rate leads to physical and exploitational interference that prevent robots from working effectively. In other words, the potential of robots to incur misplacement and opportunity costs increases with i. For example, local broadcasters, that enjoy a performance advantage over solitary swarms in most scenarios, cannot outperform them in Scatter25 when worksites are close to the base. The effect of information gain rate on swarm performance is more evident when swarms are larger, since interactions between robots are more probable.

4.4.3 Tendency to incur misplacement cost

Misplacement cost, C_M, is incurred by robots that know about a worksite but cannot obtain reward from it because they are located elsewhere. The relationship between C_M and a decrease in uncertainty cost, C_U, is expressed by the misplacement cost coefficient m (see Section 4.3.2, Equation 4.11). When $m = 0$, the swarm turns information into reward immediately. On the other hand, when $m = 1$, none of the informed robots are located at the relevant worksite and no reward is gained without travel. In the consumption task, m of solitary swarms is always 0, since they do not use recruitment and all informed robots are thus always located at their worksites. In contrast, bee recruiters, that recruit away from worksites, have the highest m in most environments.

Heap1 scenarios with a large worksite distance favour both local broadcasters and bee swarms (see Figure 4.4), despite of the fact that bee swarms have the highest m in these environments. Similarly, Heap2 and Heap4 environments favour local broadcasters over solitary swarms, even though solitary swarms do not pay misplacement cost. On the other hand, in the least difficult environments, where worksites are numerous or where the work arena is small, solitary swarms can perform well enough, while swarms that use recruitment are punished for not turning information into reward as effectively as solitary robots.

4.4.4 Tendency to incur opportunity cost

Opportunity cost, C_O, results from a tendency of a swarm to overcommit to worksites and it is paid by robots that are subscribed to worksites that have been depleted by other robots. In the consumption task, solitary swarms do not pay C_O, as robots never leave worksites until they are depleted. Bee recruiters pay an intermediate amount of C_O, since the robots recruit in the base, and thus to certain extent synchronise the time at which they arrive and deplete worksites. Local broadcasters, that recruit the whole time while they are depleting a worksite, incur the highest C_O in most environments (see Section 4.3.3).

Similarly as it is the case with misplacement cost, a strategy that can achieve a higher information gain rate but also incurs a higher opportunity cost outperforms another strategy with slower information gain rate and a lower opportunity cost in difficult environments. In the Heap1 environments, opportunity cost is irrelevant, causing both the bee swarm and local broadcasters to outperform solitary swarms. In Heap2 and Heap4 scenarios, local broadcasters pay the highest C_O, but they also pay a smaller misplacement cost than bee swarms. This leads to a better work performance, as the opportunity cost only has to be paid for a small number of worksites. However, when worksites are numerous, as it is the case in Scatter25 scenarios, robots need to spread their working effort evenly across the environment. In these cases, strategies that incur the highest C_O have a low performance.

4.4.5 The information-cost-reward framework

The relationship between characteristics of a robot control strategy and properties of the environment are complex. By measuring the ability of swarms to acquire and spread new information, as well as their tendency to incur costs when utilising that information, we can understand why certain control strategies do well in certain tasks. Environments where it is more difficult for a single robot to discover worksites favour strategies with a high information gain rate, while they allow for a certain amount of costs to be paid. On the other hand, less difficult environments do not require information gain rate to be high, and they punish swarms that pay high costs.

These relationships form the basis for the *Information-Cost-Reward (ICR) framework*, depicted in Figure 4.15. The ICR framework extends the robot work cycle (see Figure 4.11), in order to describe the work cycle of the whole swarm, as well as its relationship to the structure of the environment.

Under this framework, a swarm is understood as a single entity that acts on its environment in order to obtain reward. Reward, situated in worksites, is dispersed in the environment in a certain way, and there is a certain probability, $p(W)$, associated with

a worksite being located at a given point in space. Scouts play the role of a swarm's *sensors*. They gain information about where worksites are, decreasing the swarm's uncertainty about the environment, i.e., the amount of uncertainty cost, C_U, (Section 4.3.2, Equation 4.7) that the swarm pays. Since the swarm has new information about worksites, its expected reward, R', (Section 4.3.4, Equation 4.14) increases. Scouting success affects the swarm's information gain rate, i (Section 4.2.3, Equation 4.5). It is dependent on the difficulty of the environment, characterised by the number of worksites and the worksite distance from the base, as well on the swarm's scouting efficiency.

Upon acquiring a new piece of information, scouts can become workers, but they can also pass the information that they have to other members of the swarm in order to recruit more workers. Their ability to disperse information within the swarm depends on the swarm's communication strategy and, like scouting success, it affects the swarm's information gain rate, i. Workers act as *actuators* of the swarm and they turn the information that they have into reward, R (Section 4.3.4, Equation 4.13). However, there is a potential, unique to each control strategy, that the workers will need to incur misplacement cost, C_M, (Section 4.3.2, Equation 4.10) in order to obtain reward. Furthermore, by acting on the environment, workers eventually cause worksites to become depleted. This can result in opportunity cost, C_O, (Section 4.3.3, Equation 4.12) incurred by workers that are unable to update their information quickly enough and that are thus subscribed to non-existent worksites instead of searching for new work. At the same time, depletion of worksites increases the difficulty of the environment, causing scouts to become less successful over time.

Looking at swarm behaviour from the point of view of the ICR framework suggests how decentralised cognition emerges from actions and interactions of locally-informed embodied agents that work together to fulfil a common goal. It has been previously suggested that swarms should be understood as information-processing cognitive systems (Couzin, 2009; Trianni et al., 2011; Reina et al., 2014), similar to animal brains, where neurons process individual pieces of information but do not posses the cognitive abilities of the whole animal. The ICR framework provides a new perspective on the processes that lead to decentralised cognition by explaining the relationships between the environment, information, individual robot actions and swarm performance.

4.5 Summary

The Information-Cost-Reward framework, introduced in this chapter, portrays a swarm as a decentralised cognitive entity that perceives the environment and acts on information that it obtains. The swarm's actions, i.e., actions of individual robots that lead to emergence of global-level behaviour, are performed within a work cycle - the swarm

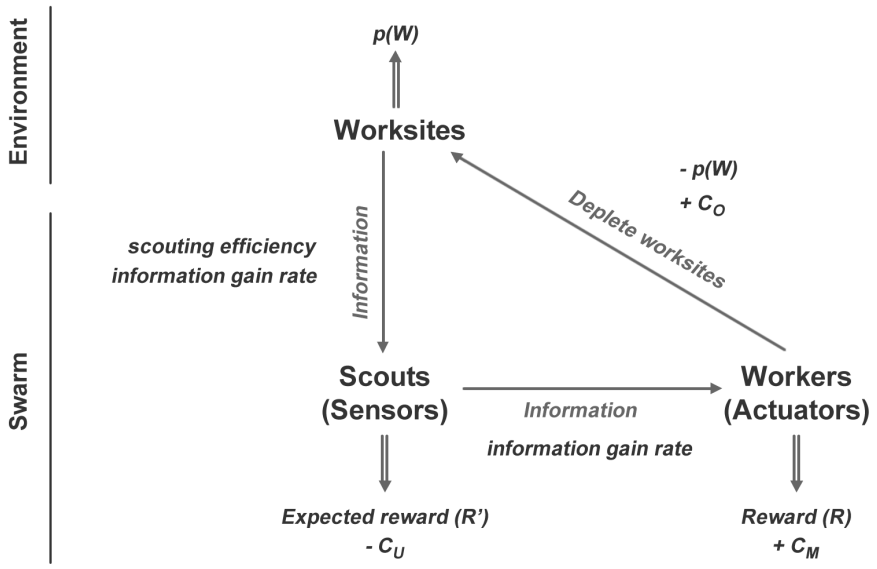


Figure 4.15: The swarm work cycle within the Information-Cost-Reward framework. Worksite distribution in the environment is characterised by the probability, $p(W)$, of a worksite being located at at a given point in space. Scouts search for and spread information about worksites. This process is affected by the swarm's scouting and recruitment strategies and produces expected reward, R', while reducing the swarm's uncertainty cost, C_U. Workers turn information into reward, R, after paying certain misplacement cost, C_M. They also alter the environment by depleting worksites, decreasing $p(W)$ and potentially increasing opportunity cost, C_O, paid by the swarm.

searches for and obtains reward from the environment, changing the environment's characteristics in turn. It does so with a certain efficiency, characterised by the costs it needs to pay when turning information into reward.

Difficult environments, where worksite density is low, favour strategies with a high information gain rate, i. However, since a high information gain rate cannot be achieved without communication between robots, having a higher i leads to higher costs incurred while converting information into reward, as a result of spatio-temporal dissociation of robots and worksites. Consequently, in less difficult environments, where worksites are easy to find, strategies that have a lower i but also incur lower costs perform better.

Using the ICR framework, we can form hypotheses about how swarms, characteristics of which are known to us, will perform different tasks. In the next chapter, the collection task in static environments is investigated. During collection, the environment itself creates a misplacement of reward from worksites, as robots need to return collected items to a drop off location. It is argued that a certain amount of misplacement cost is thus ameliorated by the nature of the task. Consumption and collection tasks in dynamic environments are investigated in Chapters 6 – 8. A dynamic environment, where worksites change their location periodically, creates a stronger pressure on swarms that pay opportunity cost and thus changes the relative performance of swarms. Finally,

design patterns for robot swarms are described by using the terminology of the ICR framework in Chapter 9.

Chapter 5

Misplacing the reward: Collection in static environments

The results presented in the previous chapter show that different robot control strategies are suitable in different environments when performing the consumption task. According to the Information-Cost-Reward (ICR) framework, introduced in Section 4.4.5, the ability of robots to acquire and share information, as well as their potential to incur misplacement and opportunity costs when using that information, determine how suitable their control strategy is in a given environment.

In this chapter, the collection task in static environments is investigated, using the three basic control strategies (solitary, local broadcaster and bee swarms). The ICR framework is utilised in order to describe and explain the performance of these swarms.

In the collection task, a robot picks up a pellet from a worksite and delivers it back to the base, where it obtains a reward. Robots have to perform a number of round trips between a worksite and the base in order to fully deplete it. In the terminology of the ICR framework, a robot needs to periodically *misplace* itself away from its worksite in order to gain reward. This has two important consequences:

1. The time required to deplete a worksite increases with worksite distance from the base
2. There is a possibility that a worksite will be depleted by other robots while a worker is making a trip to the base

The cost analysis presented in Section 4.3 suggests that poorly managing misplacement cost is one of the reasons why some swarms convert information into reward ineffectively. In the consumption task discussed in the previous chapter, bee swarms are especially disadvantaged, as a bee scout returns to the base to recruit before it begins extracting

reward from a worksite that it found, incurring misplacement cost. In this chapter, it is hypothesised that:

Hypothesis 5.1. The effect of misplacement cost on a strategy that utilises the base for information exchange is ameliorated in the collection task, since the task itself requires the robots to travel to the base.

If the hypothesis holds, the bee swarm strategy should outperform the other control strategies, because its high misplacement cost is associated with a high information gain rate. Sharing information about where worksites are located is advantageous to the robots. When the associated misplacement cost becomes less relevant, as is the case in the collection task, a control strategy with a high information gain rate should perform better than other strategies that have a lower information gain rate.

The four swarm characteristics (scouting efficiency, information gain rate and the tendency to incur misplacement and opportunity costs) in the collection task are first examined. The performance of the swarms is then compared in terms of the task completion time. It is demonstrated that Hypothesis 5.1 is supported in most Heap environments. On the other hand, since the performance of swarms is also dependant on the opportunity cost incurred, bee swarms have a lower performance than the other swarms in environments with a high worksite density and small worksite volumes.

5.1 Swarm characteristics

5.1.1 Scouting efficiency

In the collection task, the impact of the number of worksites, N_W, and worksite distance, D, on the scouting efficiency of swarms was similar as in the consumption task (see Section 4.2.1). Bee swarms suffered from a deterioration in scouting efficiency that was

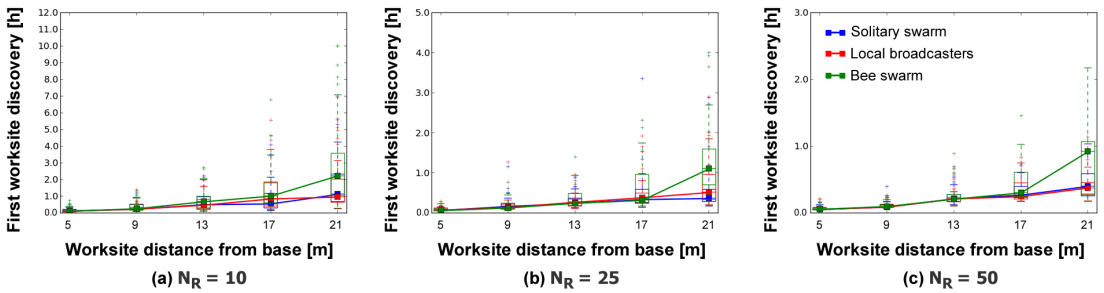


Figure 5.1: Time of the first worksite discovery in the static collection task, Heap1 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, $D = 21$m, was larger in bee swarms, compared to the other swarms.

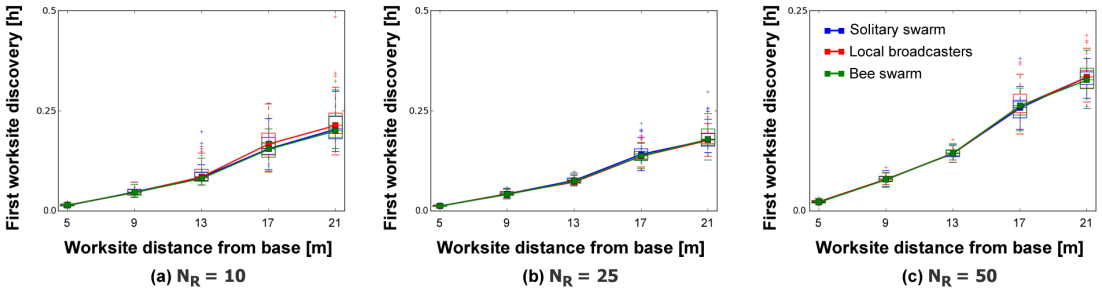


Figure 5.2: Time of the first worksite discovery in the static collection task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was similar for all swarm types.

significantly larger than that of other swarms in the Heap environments with a large D (see Figure 5.1 and Appendix D, Figures D.1 and D.2). As in the consumption task, the poor scouting efficiency of bee swarms in these environments was caused by the fact that bee swarm scouts periodically returned to the base in order to check whether other robots had information about worksites to share.

On the other hand, in Scatter25 environments, where N_W was large, the scouting efficiency of all swarms was affected by D in a similar fashion (see Figure 5.2).

5.1.2 Information gain rate

The information gain rate, i, (see Section 4.2.3) of the swarms was similar to that in the consumption task. Bee swarms, that utilised a designated place for recruitment, enjoyed the highest information gain rate among the swarms in most of the Heap environments, followed by local broadcasters and solitary swarms (see Figure 5.3 and Appendix D, Figures D.3 and D.4).

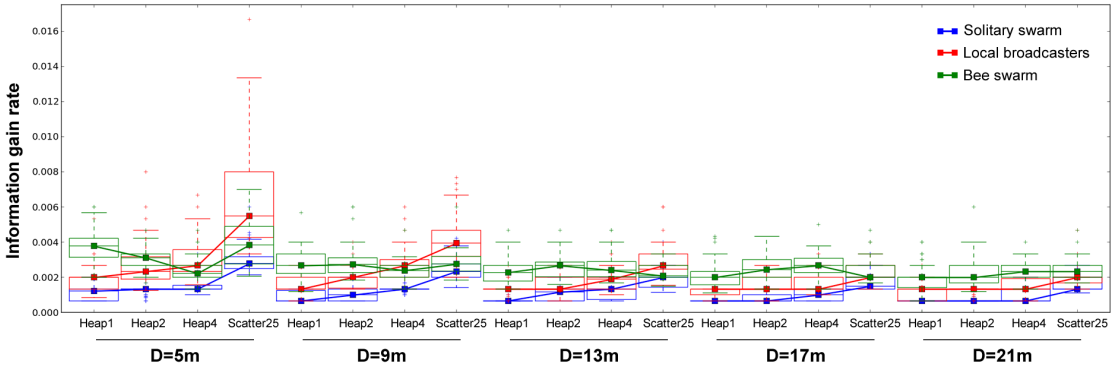


Figure 5.3: Information gain rate, i, of 25-robot swarms in the static collection task. Bee swarms enjoyed the highest i in most environments, apart from the Scatter25 scenarios when $D \leq 13$m and Heap4 scenarios when $D \leq 9$m.

In contrast, in Scatter25 and Heap4 environments with a small D, local broadcasters achieved the highest i. In these environments, the high density of worksites and robots favoured the recruitment behaviour of local broadcasters, since it was very probable that robots would find themselves close to each other.

The information gain rate was always the lowest in solitary swarms, since solitary robots did not share information with each other.

5.1.3 Tendency to incur misplacement cost

Tendency to incur misplacement cost, C_M, (see Section 4.3.2) was the most affected swarm characteristic in the collection task. Since the robots had to make round trips between the base and worksites, they paid C_M most of time when they were subscribed to a worksite. Consequently, the median misplacement cost coefficient, m, (see Section 4.3.2) of all swarms was equal to 1 in all scenarios and for all swarm sizes (see Figure 5.4 and Appendix D, Figures D.5 and D.6).

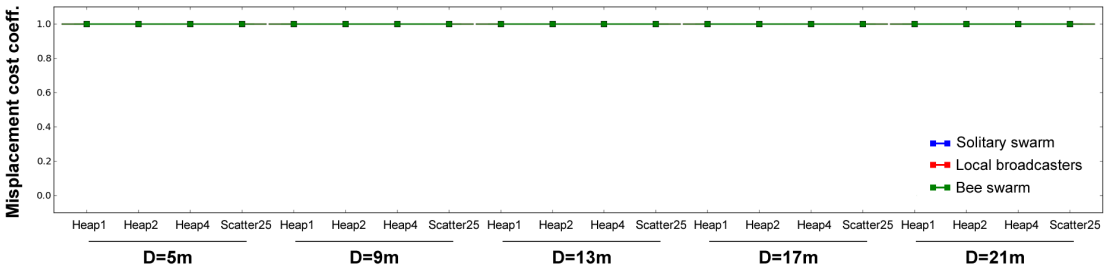


Figure 5.4: Misplacement cost coefficient, m, of 25-robot swarms in the static collection task. Because robots needed to travel to the base to obtain reward, m of all swarms was equal to 1.

5.1.4 Tendency to incur opportunity cost

Unlike in the consumption task, paying some amount of opportunity cost, C_O, (see Section 4.3.3) was unavoidable in the collection task, regardless of the swarm type (see Figure 5.5 and Appendix D, Figures D.7 and D.8). Since robots periodically left worksites and travelled to the base, it was possible for the worksites to get depleted by other robots. The tendency to incur C_O was the lowest in solitary swarms. Solitary robots did not recruit, meaning that there was a smaller probability for multiple workers to subscribe to the same worksite, compared to local broadcasters and bee swarms.

Bee swarms paid the highest C_O in all environments, since the robots spent more time away from worksites than robots from other swarms. Apart from traveling to the base and back in order to obtain reward, bee robots spent time in the base recruiting after each round trip.

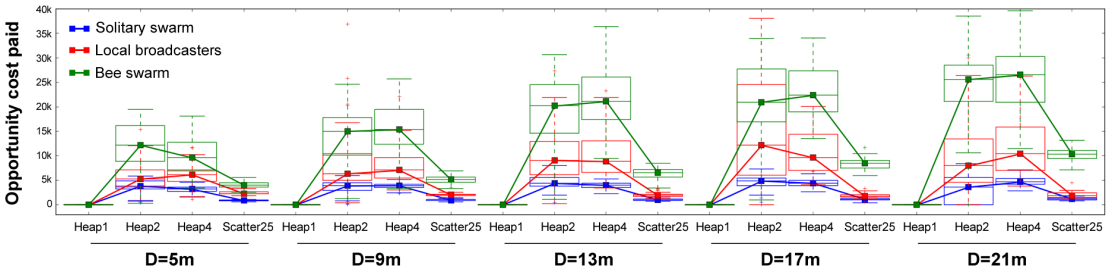


Figure 5.5: Opportunity cost, C_O, of 25-robot swarms in the static collection task. Bee swarms paid the largest amount of C_O, followed by local broadcasters and solitary swarms. C_O of all swarms increased with worksite distance, D, given the same scenario type.

Local broadcasters, that recruited around worksites while extracting resource from them, paid an intermediate level of C_O in all environments.

Finally, it is notable that the opportunity cost incurred by robots of all swarm types was higher when worksite distance was larger. This was caused by the fact that a robot's round trip to the base took longer when a worksite was further away from the base, increasing the amount of time during which other robots could deplete the worksite, as well as the time that it took the robot to travel back to the worksite and realise that it is no longer active.

5.2 Swarm-environment fit

During the collection task, robots need to periodically return to the base in order to unload resource. This causes robots to become misplaced from worksites, and increases the probability that the worksites will get meanwhile depleted by other members of the swarm. Consequently, the tendency of control strategies to incur costs during work is different than that in the consumption task. Firstly, the misplacement cost, C_M, is paid during most of the robot's work cycle. Secondly, the tendency of robots to incur opportunity cost, C_O, is higher than in the consumption task, and increases with worksite distance from the base (i.e., with duration of a foraging trip). All swarms pay some amount of C_O, although the cost is generally smaller for swarms with a lower information gain rate.

While the tendency to incur costs is different than in the consumption task, the information flow characteristics of swarms, i.e., the scouting efficiency and the information gain rate, are similar. This is because the structure of the environment, characterised by a probability, $p(W)$, of a worksite being located at a given point in space, is the same as in the consumption task. As worksite distance increases, bee swarms suffer from deterioration of scouting efficiency and this deterioration is more severe than that

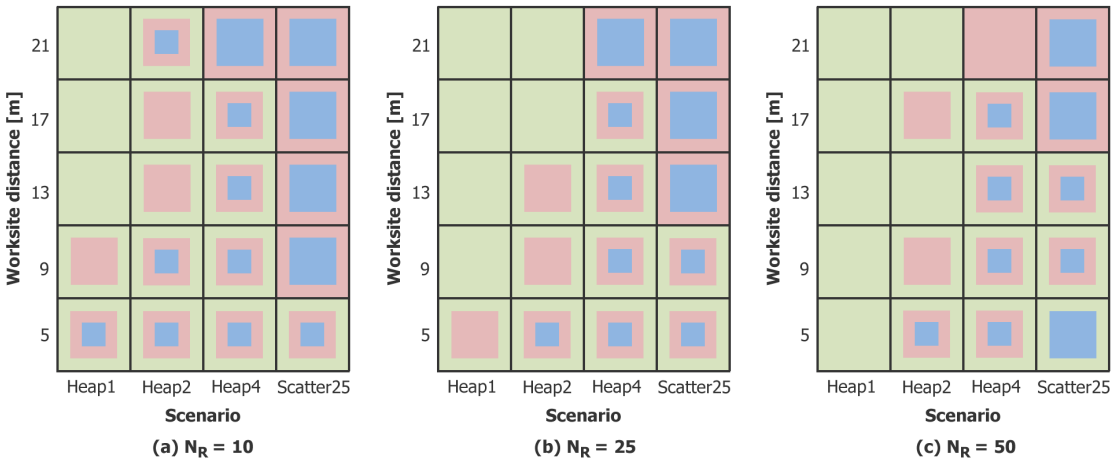


Figure 5.6: Winning strategies that completed the static collection task the fastest in different environments using (a) 10, (b) 25 and (c) 50 robots. The strategies are identified by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix D, Figures D.9 - D.11.

suffered by other swarms. However, because they exchange information in the base, bee swarms usually have the highest information gain rate, followed by local broadcasters and solitary swarms.

Hypothesis 5.1 predicts that bee swarms should outperform other swarms in the static collection task, because their misplacement cost is ameliorated by the fact that the robots need to travel to the base to unload resource. This hypothesis is supported in the Heap environments (Figure 5.6), most notably in Heap1. In these environments, finding worksites is difficult and bee swarms cope with this difficulty by achieving a high information gain rate by exchanging information in the base.

On the other hand, bee swarms cannot outperform the other swarms in easy environments, where the number of worksites is high ($N_W = 25$) or when worksites are close to the base ($D = 5$m). Similarly as it is the case in the consumption task, solitary swarms perform better than other swarms in the Scatter25 scenario when $D = 5$m and share a similar performance with local broadcasters in the other easy environments.

This result can be explained by using the ICR framework. Apart from information gain rate and misplacement cost, opportunity cost affects the swarm performance as well. Bee swarm robots incur the highest C_O, followed by local broadcasters and solitary robots. In the least difficult environments, where the swarm needs to spread out in order to explore and exploit the environment effectively, the amount of C_O paid is more important than the information gain rate, i. Exploitational interference, that manifests itself as a high amount of C_O paid, prevents the bee swarms from outperforming the

other swarms, despite the fact that bee swarm robots enjoy a higher i and their C_M is ameliorated.

5.3 Summary and discussion

The bee swarm strategy, which utilises the base as a place where robots meet and exchange information about worksites, is favoured in the collection task in environments where it is difficult for robots to find worksites on their own. This result is different from that in the consumption task.

In the consumption task, bee swarms are disadvantaged since they spend an unnecessary amount of time traveling to the base in order to recruit, generating a high misplacement cost. However, in the collection task, where reward can only be obtained in the base, i.e., away from worksites, the misplacement cost of bee swarms is ameliorated by the nature of the task. This allows the bee swarms to take an advantage of the high information gain rate that they achieve.

On the other hand, in less difficult environments, where work sites are numerous or very close to the base, the high amount of opportunity cost incurred by the bee swarms prevents them from outperforming other swarms with a lower information gain rate, such as local broadcasters and solitary swarms.

Foraging robot swarms, similar to the solitary and bee swarms investigated here, were previously compared in a Java-based simulation environment (Pitonakova et al., 2014, see Appendix H). In line with the results presented in this chapter, it was shown that swarms that do not utilise communication outperform bee-inspired swarms in environments where a lot of small item deposits are randomly dispersed in the environment, but that central-place recruitment provides an advantage in environments where deposits are difficult to find. Furthermore, when robots communicate worksite locations to each other, errors caused by sensory noise accumulate quicker than in solitary swarms. Other authors (e.g. Krieger and Billeter, 2000; Gutiérrez et al., 2010; Lee et al., 2013) performed similar experiments, both in simulation and in the real world, and demonstrated similar results.

Chapter 6

Missing opportunities: Working in dynamic environments

In statics tasks, explored in Chapters 4 and 5, worksite locations were determined at the beginning of each simulation run and remained constant. A task was considered completed when all reward was extracted from the environment. The static consumption and collection tasks are a model for real-world tasks with clearly predefined goals, for example, cleaning of a particular street or collection of a certain amount of minerals from the environment. However, a robot swarm might be required to perform a continuous task instead, such as picking up and delivering packages, or maintaining a set of machines in a warehouse by responding to machine breakdowns. In such applications, locations at which work needs to be performed would change over time. In order to better understand the control strategies investigated previously, it is important to test them in dynamic tasks as well.

In this chapter, dynamic swarm tasks are modelled by changing worksite locations every T_C simulated hours. All active worksites are removed from the environment at the end of each change interval, and new undepleted worksites are added to the environment. The new worksite locations are random, but constrained by the rules of a particular scenario type. For example, in the Heap4 scenario with $D = 9$m, four new worksites, each having $100/4 = 25$ units of volume, replace four old worksites at the end of each T_C interval. The new worksites are placed at random positions 9m away from the base edge (Figure 6.1). The performance of swarms is evaluated in terms of the *total* amount of reward collected during 50 independent experimental runs. Each run is T hours long, where T depends on swarm size (see Table 6.1).

Dynamic consumption and collection tasks are explored. There are two versions of each dynamic task, a *slow* and a *fast* version, corresponding to different values of T_C. The values of T_C and T are listed in Table 6.1 and are determined as follows. First, the average time, T', in which swarms of all three types and of a particular size extract

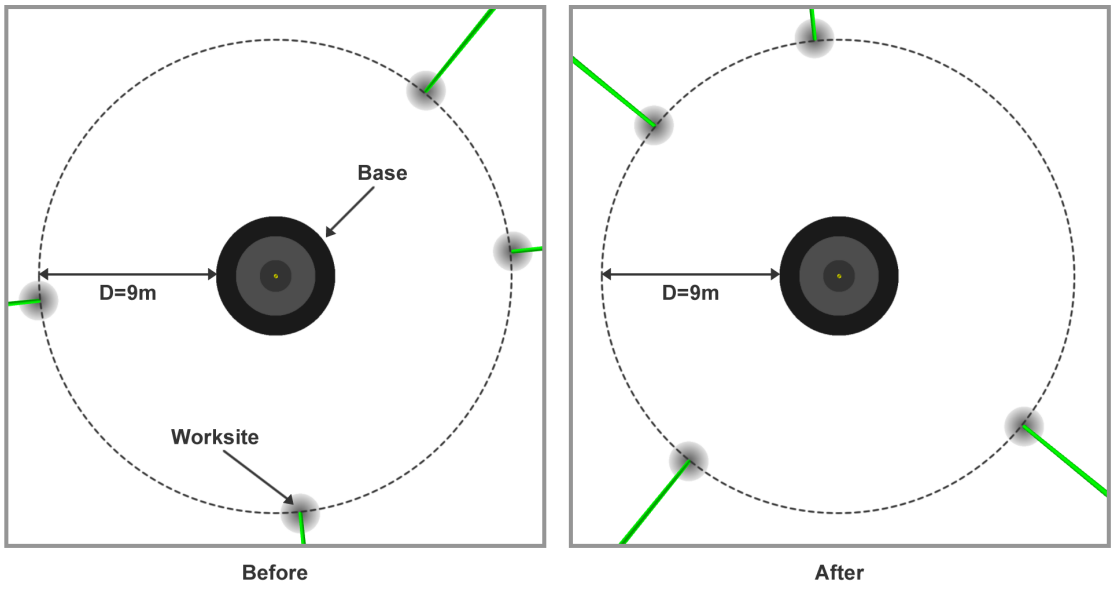


Figure 6.1: The Heap4 scenario with $D = 9$m, before and after the environment changed at the end of a T_C interval. The central base is surrounded by four worksites, each 9m away from the base edge. When the environment changes, four new random worksite locations, 9m away from the base edge, are generated.

N_R	T' **in static tasks**	T_C **in slow dynamic tasks**	T_C **in fast dynamic tasks**	T **in dynamic tasks**
10	6673s	6300s	3150s	17.5h
25	3813s	3600s	1800s	10h
50	2615s	2700s	1350s	7.5h

Table 6.1: Values of T_C and T selected for different swarm sizes based on the average time, T', in which swarms of each size extract half of the reward from the environment in both static collection and consumption tasks.

half of the reward from the environment, in all 20 investigated scenarios (Heap1, Heap2, Heap4 and Scatter25, each with $D \in \{5, 9, 13, 1, 7, 21\}$m), in both static collection and consumption tasks, in measured. The value of T_C in *slow* dynamic tasks is set to a multiple of 15 minutes (900 seconds) that is close to T', allowing swarms to deplete on average half of the worksites before the environment changes[1]. In *fast* dynamic tasks, the value of T_C for a particular swarm size is set to half of that used in slow dynamic tasks. The total length of a simulation run, T, is the same in slow and fast dynamic tasks and it is equal to ten times the value of T_C in slow dynamic tasks. That is, the environment changes 10 times in slow dynamic tasks and 20 times in fast dynamic tasks.

[1]However, since the performance of swarms differs depending on the control strategy used, the task type, the number of worksites and the worksite distance from the base, a different number of worksites is depleted in each particular scenario.

In dynamic tasks, reward from a particular worksite is only available for a limited amount of time. This has the following implications:

- Robots need to discover new worksites every time the environment changes
- Reward needs to be extracted from the discovered worksites as soon as possible
- Overcommitment to a small portion of the currently active worksites results in lost opportunity to obtain other, temporarily available, reward

It is thus hypothesised that:

Hypothesis 6.1. Scouting behaviour of a control strategy is more important in dynamic environments as opposed to static environments. The scouting efficiency of controllers differs more in dynamic environments.

Hypothesis 6.2. A strategy that uses information about worksites ineffectively in a static task, i.e., has an unnecessarily high misplacement coefficient, m, performs significantly worse than other control strategies in a dynamic version of that task.

Hypothesis 6.3. A control strategy with a lower tendency to incur C_O enjoys a performance benefit over other strategies in more scenarios when a task is dynamic, compared to when it is static.

The first hypothesis predicts that scouting efficiency of a strategy that exhibits ineffective scouting behaviour will be lower than that of other strategies more often in dynamic, compared to static environments. If the hypothesis holds, the scouting efficiency of bee swarms should be negatively affected by worksite distance more severely than that of local broadcasters and solitary swarms in dynamic environments.

The second hypothesis is related to the way in which information is shared between robots. In the consumption task, robots should not leave worksites once they discover them, as reward can be extracted from the worksites immediately. However, bee robots incur a large amount of misplacement cost by attempting to recruit in the base. It is thus expected that bee swarms, that perform similarly to other swarms in some environments during the static consumption task, will perform significantly worse than other swarms in the dynamic version of the task.

Finally, according to the third hypothesis, solitary swarms, that pay the lowest amount of C_O in both consumption and collection tasks, should outperform other swarms in a larger number of scenarios when environments are dynamic, compared to when they are static. By extension, bee swarms, and in some scenarios local broadcasters, that pay the highest amount of C_O in the static collection task, should loose their advantage over other swarms when the collection task is dynamic.

In the following sections, the swarm characteristics are examined and the performance of swarms is compared within the context of the ICR framework. It is shown that Hypothesis 6.1 holds in all environments and in both the consumption and the collection task. Secondly, as it is predicted by Hypothesis 6.2, bee swarms perform significantly worse than other swarms in the dynamic consumption task. Finally, while Hypothesis 6.3 is supported in most cases, there are some scenarios where local broadcasters, even though they pay the highest amount of C_O, outperform the other control strategies. In these cases, a high amount of C_O incurred by local broadcasters is less important than their ability to share information quickly and their low tendency to incur misplacement cost.

6.1 Swarm characteristics

6.1.1 Scouting efficiency

Scouting efficiency in static environments was evaluated in terms of the time that it took a swarm to discover the first worksite (see Section 4.2.1). In dynamic environments, the time of the first worksite discovery in each change interval is measured and the scouting efficiency is evaluated based on the *median* first worksite discovery time when all intervals are considered. If no worksite discovery is made during a change interval, the first worksite discovery time is recorded as the maximum possible value, i.e., the length, T_C, of the change interval.

The ability of all swarms to discover new worksites was lower when worksite distance was large, most significantly for bee swarms, while local broadcasters and solitary swarms were affected similarly to each other (see Figures 6.2 – 6.5). Compared to static environments, all controllers were affected by worksite distance by a larger amount, since the robots needed to repeatedly discover new worksites. Moreover, scouting efficiency was

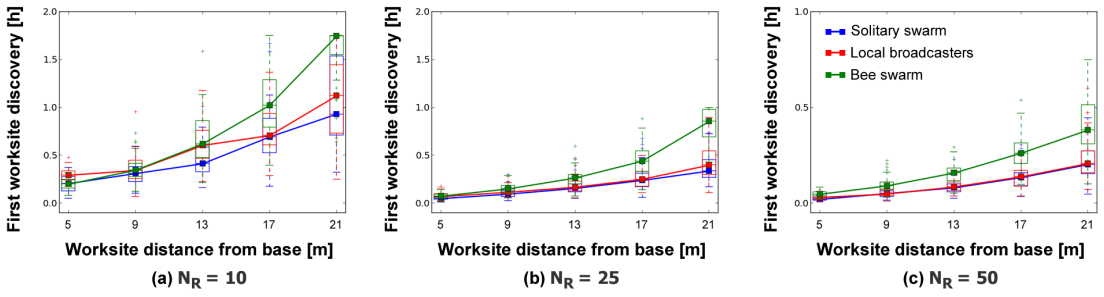


Figure 6.2: Time of the first worksite discovery in the slow dynamic consumption task, Heap1 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other control strategies.

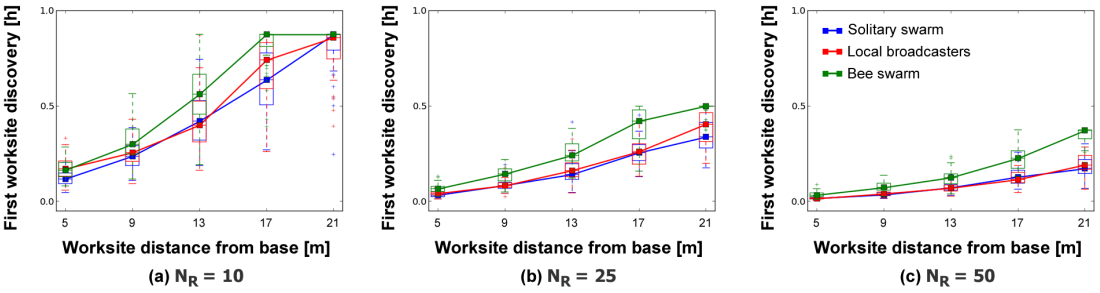


Figure 6.3: Time of the first worksite discovery in the fast dynamic consumption task, Heap1 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other control strategies. Due to the limited time available in each change interval, the swarms were usually unable to discover the worksite when $D = 21$m and $N_R = 10$. Bee swarms of 10 robots were also usually unable to discover the worksite when $D = 17$m.

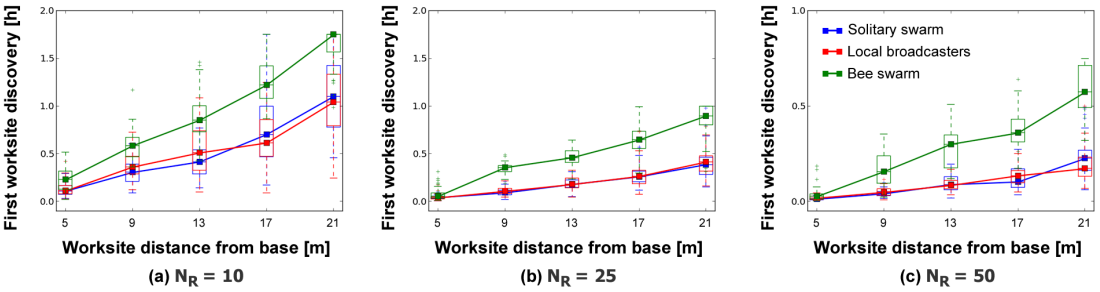


Figure 6.4: Time of the first worksite discovery in the slow dynamic collection task, Heap1 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other control strategies.

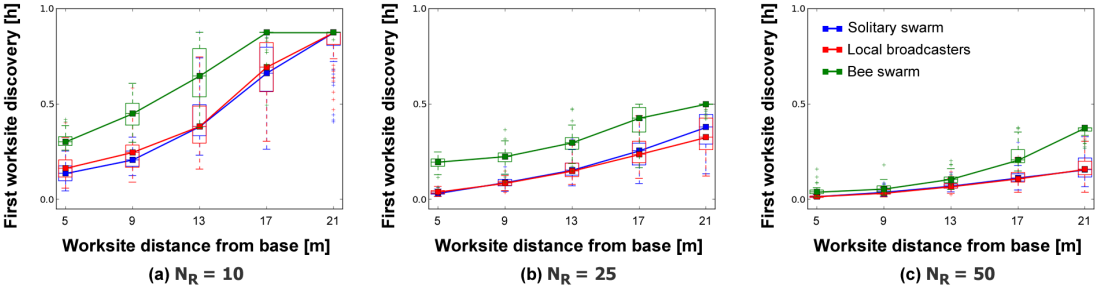


Figure 6.5: Time of the first worksite discovery in the fast dynamic collection task, Heap1 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other control strategies. Due to the limited time available in each change interval, the swarms were usually unable to discover the worksite when $D = 21$m and $N_R = 10$. Bee swarms of 10 robots were also unable to discover the worksite when $D = 17$m.

negatively affected by worksite distance more strongly when the environment changed quickly (e.g., compare Figures 6.2 and 6.3), and when swarm size was small (e.g., compare Figures 6.2a and 6.2b). In contrast, scouting efficiency was affected less strongly when there were more worksites available in the environment (see Appendix E, Figures E.1 – E.8).

It is notable that bee swarms took longer than other swarms to discover new worksites in almost all scenarios, regardless of worksite distance. This result is different from that in static environments, where bee swarms took longer than the other swarms only when worksite distance, $D = 21$m. The scouting strategy of bee robots, which periodically returned to the base to check whether other robots had any new information, prevented the bee swarms from sampling the environment as often as the other swarms did. This was a problem when the environment changed periodically and needed to be explored continuously.

6.1.2 Information gain rate

The information gain rate, i, (see Section 4.2.3) of 25-robot solitary swarms was higher in dynamic environments, compared to static environments (Figure 6.6). Since the robots were already dispersed in the work arena when new worksites were added to it, the robots were more likely to discover the new worksites quickly at the beginning of a new

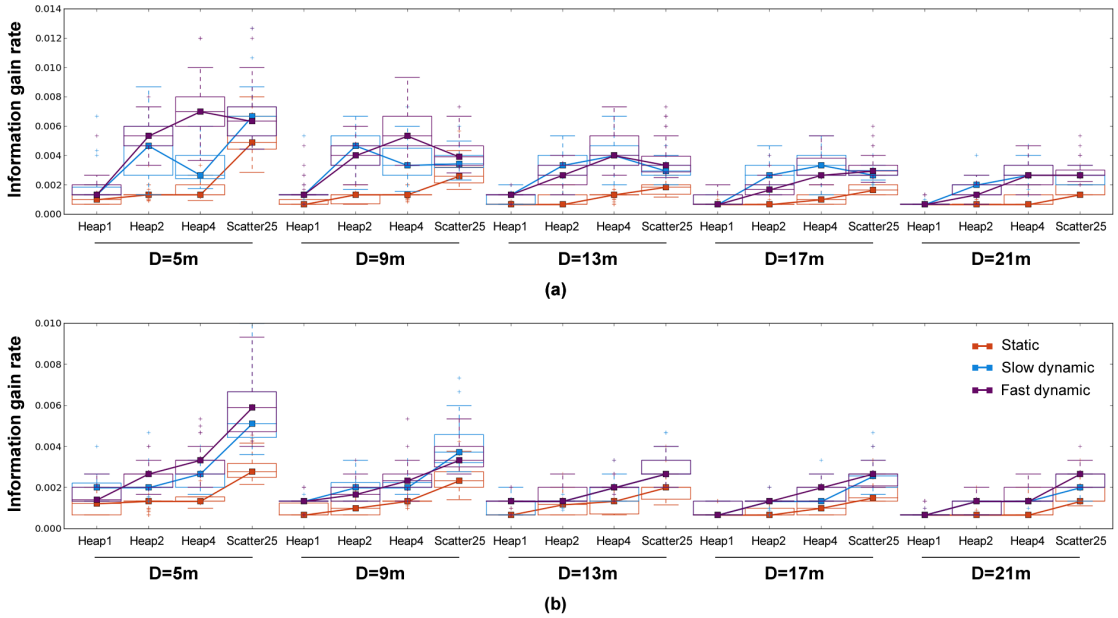


Figure 6.6: Information gain rate, i, of 25-robot solitary swarms in the (a) consumption, (b) collection task. The information gain rate was higher in dynamic compared to static environments, most notably in the consumption task or when worksite distance, D was small.

T_C interval, compared to the beginning of a simulation run in static environments. This was especially true in the consumption task, where the robots did not have to travel to the base to unload resource. The information gain rate was the highest when worksite density was large, i.e., when N_W was large or when D was small. Solitary swarms of 10 and 50 robots showed a similar increase in i (see Appendix E, Figures E.9 and E.10).

The information gain rate of local broadcasters was also higher in dynamic, compared to static environments and the increase in i showed similar patterns to those observed in solitary swarms (see Figure 6.7 for results from 25-robot swarms and Appendix E, Figures E.11 and E.12 for results from 10- and 50- robot swarms, respectively).

Bee swarms exhibited the lowest increase in i in both the consumption and the collection task (see Figure 6.8 for results from 25-robot swarms and Appendix E, Figures E.13 and E.14 for results from 10- and 50- robot swarms, respectively). The lack of increase in i was caused by the inability of bee swarms to discover new worksites quickly, due to their ineffective scouting strategy (see Section 6.1.1).

When environments were static, the information gain rate of bee swarms was higher than that of other controllers in most cases during both the consumption and the collection task (see Figures 4.10 and 5.3). However, in dynamic environments, this was rarely true. In the dynamic consumption task, i of local broadcasters was the highest in less difficult environments, i.e., when N_W was high or D was small, and similar to that of bee swarms in other environments (see a comparison between 25-robot swarms in Figure 6.9 and a

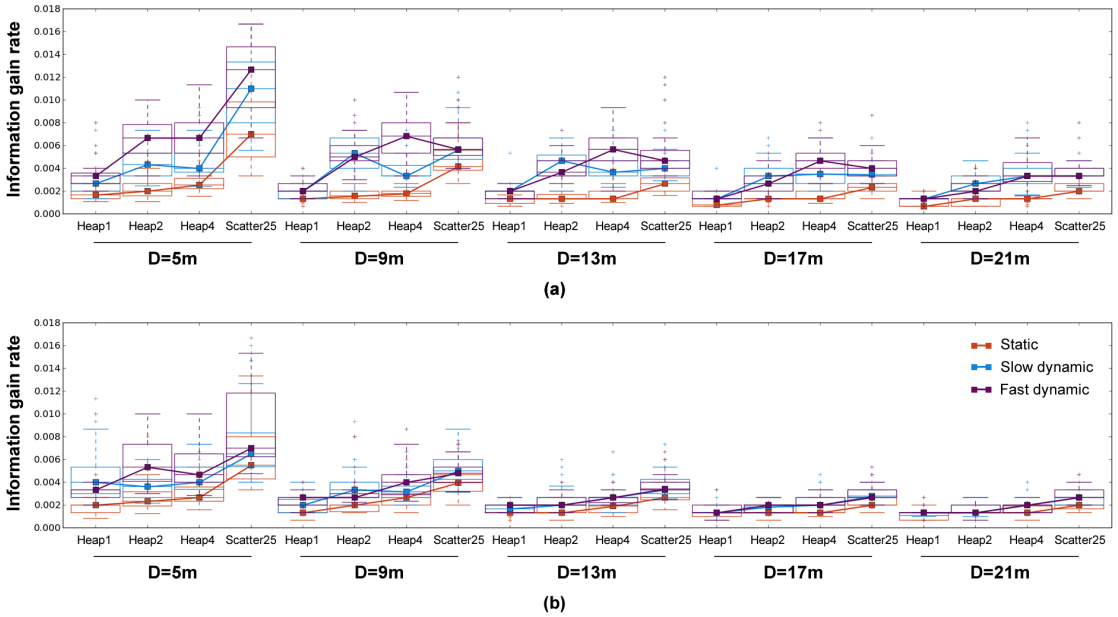


Figure 6.7: Information gain rate, i, of 25-robot local broadcaster swarms in the (a) consumption, (b) collection task. The information gain rate was higher in dynamic compared to static environments, most notably in the consumption task or when worksite distance, D was small.

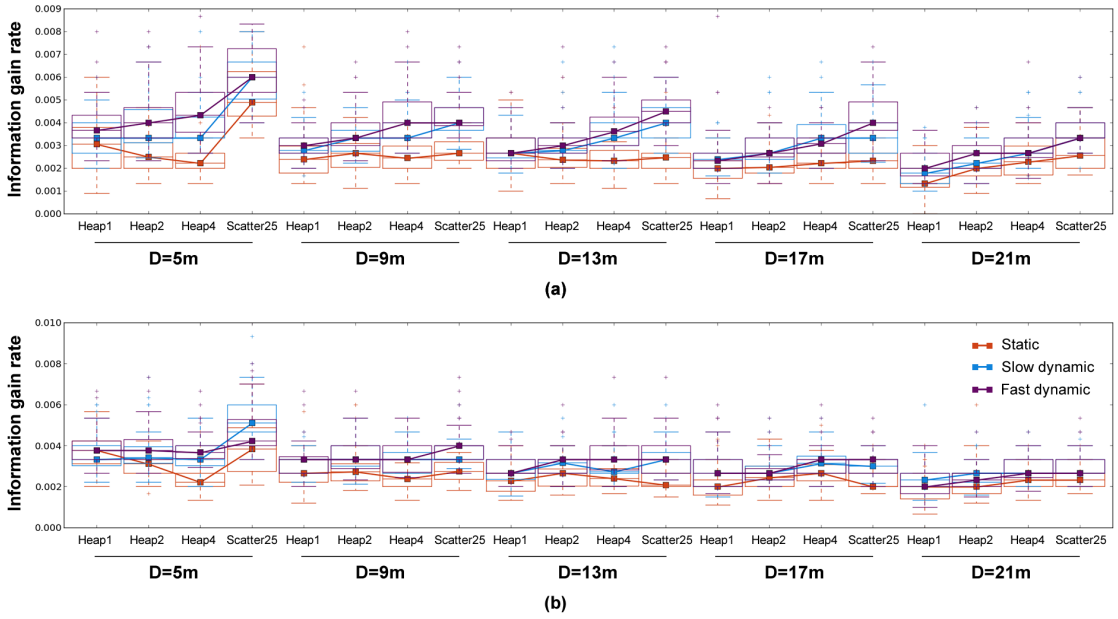


Figure 6.8: Information gain rate, i, of 25-robot bee swarms in the (a) consumption, (b) collection task. The information gain rate was higher in a small number of dynamic, compared to static environments, most notably in the consumption task or when worksite distance, D was small.

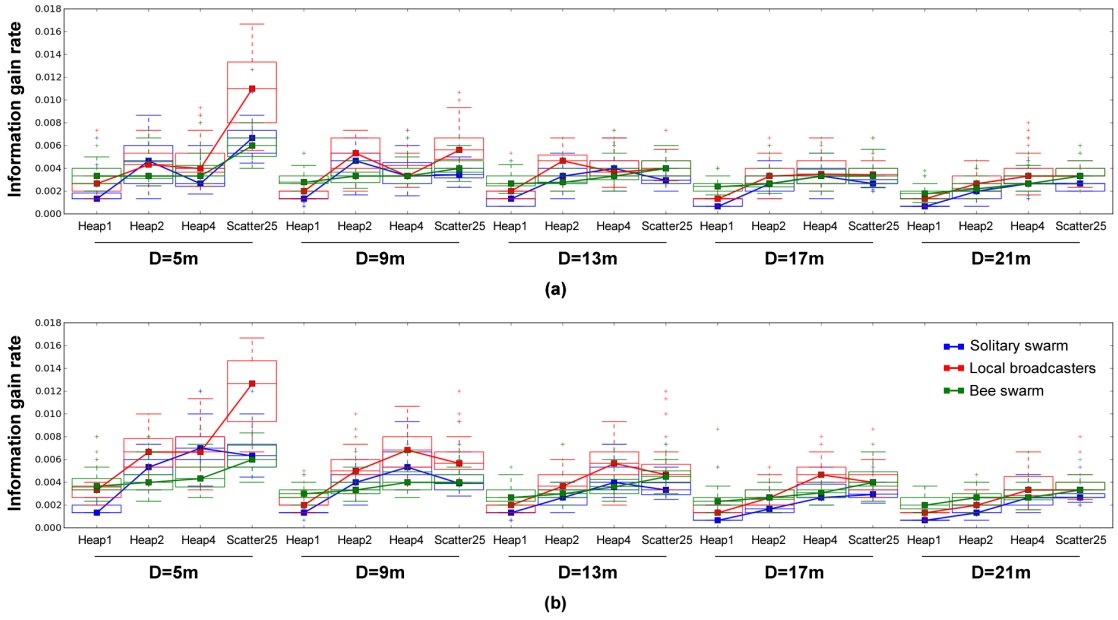


Figure 6.9: Information gain rate, i, of 25-robot swarms in the (a) slow and (b) fast dynamic consumption task. Bee swarms enjoyed the highest i in Heap1 scenarios. Local broadcasters had the highest i in most scenarios when the environmental change was fast and $D \leq 13$m, while they shared a similar i with solitary and bee swarms in other cases.

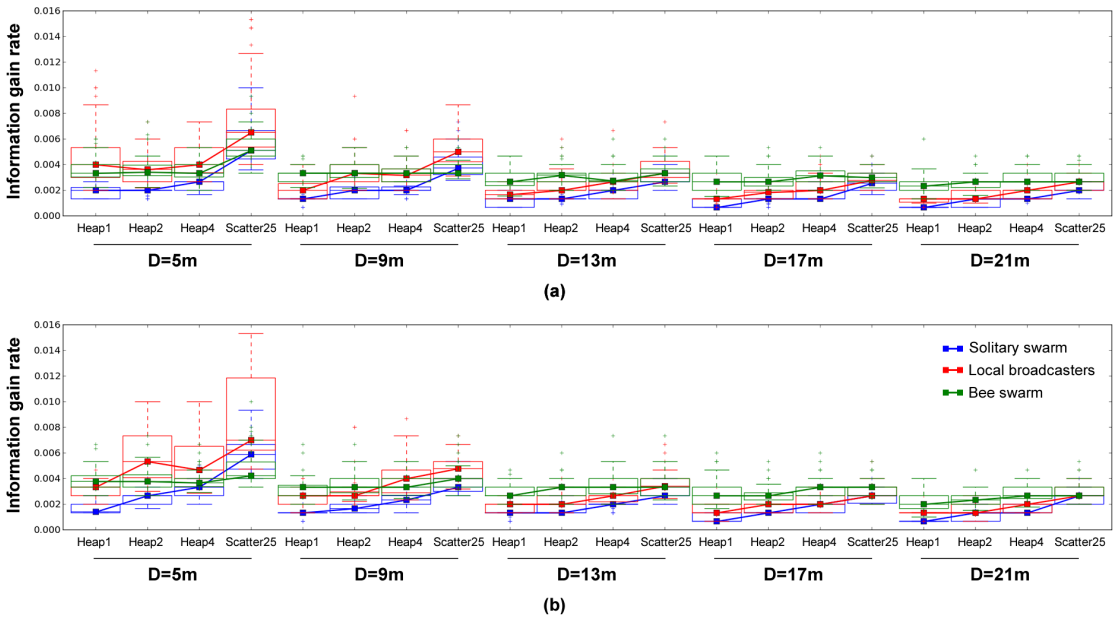


Figure 6.10: Information gain rate, i, of 25-robot swarms in the (a) slow and (b) fast dynamic collection task. Bee swarms enjoyed the highest i in most scenarios. Local broadcasters had the highest i when the number of worksites was large or when D was small.

comparison between 10- and 50-robot swarms in Appendix E, Figures E.15 and E.16). The advantage of local broadcasters over bee swarms was higher when the environment change was fast, i.e., when it was more important to continuously scout the environment. However, bee swarms retained the highest i in the most difficult, Heap1 scenarios.

In the dynamic collection task, where robots needed to travel to the base in order to unload resource, the comparison between controllers was more similar to that in static environments. Bee swarms usually achieved the highest i, followed by local broadcasters and solitary swarms (see a comparison between 25-robot swarms in Figure 6.10 and a comparison between 10- and 50-robot swarms in Appendix E, Figures E.17 and E.18). Local broadcasters had the highest i in the least difficult environments, i.e., when N_W was high and when D was small, as the robots were usually close to each other and could thus communicate more frequently than bee robots.

6.1.3 Tendency to incur misplacement cost

It was shown in Chapters 4 and 5 that misplacement cost, C_M, (see Section 4.3.2) is incurred differently in the consumption, compared to the collection task. In the consumption task, C_M is only paid by strategies that utilise recruitment, as recruits usually need to travel to the worksite upon learning about it. In contrast, all robots pay C_M in the collection task, since they need to travel between the base and worksites in order to obtain reward. It follows that the median misplacement cost coefficient, m, (see

Section 4.3.2) of swarms is a real number between 0 and 1 in the consumption task. In the collection task, the median $m = 1$. In this section, only the m in the consumption task is thus investigated. Furthermore, since solitary robots do not recruit, and thus do not pay misplacement cost in the consumption task, only local broadcasters and bee swarms are discussed here.

Misplacement cost is incurred by robots that know about an active worksite but are not located at it. A robot stops incurring misplacement cost if:

(a) It reaches a worksite that it is subscribed to (i.e., a worksite that the robot has decided to perform work at)

(b) The worksite that the robot is subscribed to is removed from the environment

In static environments, a worksite can only be removed by the robots, i.e., when it is depleted. However, in dynamic environments, a worksite can also be removed as a result of environmental change, i.e. at the end of each T_C interval. Therefore, the values of m were *lower* in dynamic, compared to static environments when local broadcasters were used in the Heap1 scenarios with a small D (see Figure 6.11 for results from 25-robot swarms and Appendix E, Figures E.19 and E.20 for results from 10- and 50- robot swarms, respectively). In these cases, recruited robots were likely to incur misplacement cost as a result of congestion near the worksite. When the environment changed, these robots stopped incurring C_M and started incurring opportunity cost instead (see Section 6.1.4 below), which decreased the misplacement coefficient of the swarm. In other scenarios, where congestion did not play a significant role, the m of local broadcasters was similar in dynamic and static consumption tasks.

The misplacement cost coefficient of bee swarms was affected more severely than that of local broadcasters (see Figure 6.12 for results from 25-robot swarms and Appendix E, Figures E.21 and E.22 for results from 10- and 50- robot swarms, respectively). Their m was usually higher in dynamic, compared to static environments, especially when

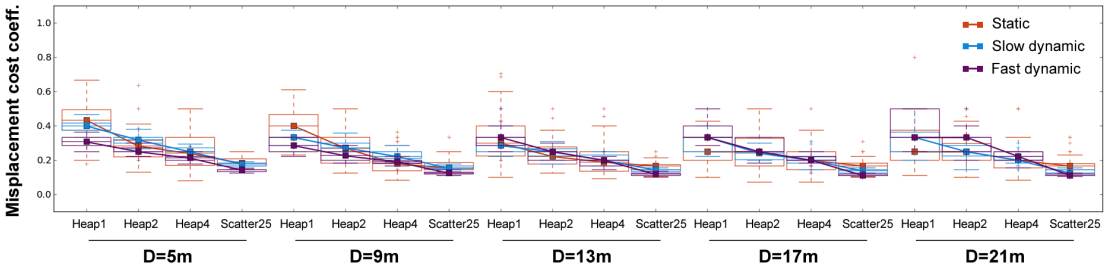


Figure 6.11: Misplacement cost coefficient, m, of 25-robot local broadcaster swarms in the consumption task. The value of m was similar in static and dynamic tasks in most environments. However, m was smaller in dynamic, compared to static, Heap1 scenarios when $D \leq 9$m.

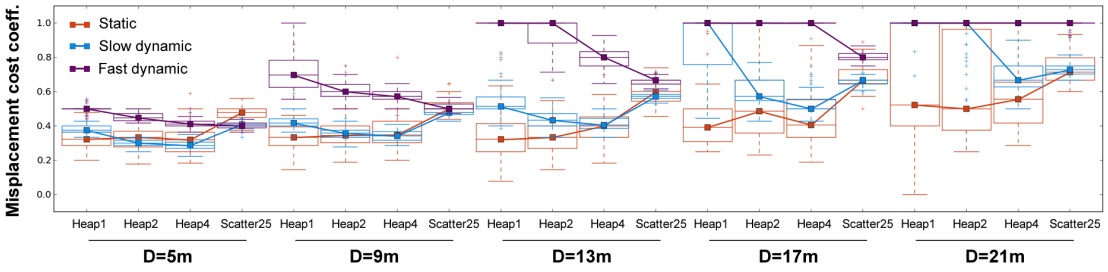


Figure 6.12: Misplacement cost coefficient, m, of 25-robot bee swarms in the consumption task. The value of m was higher in dynamic, compared to static environments, most notably when worksite distance was large or when the environmental change was fast.

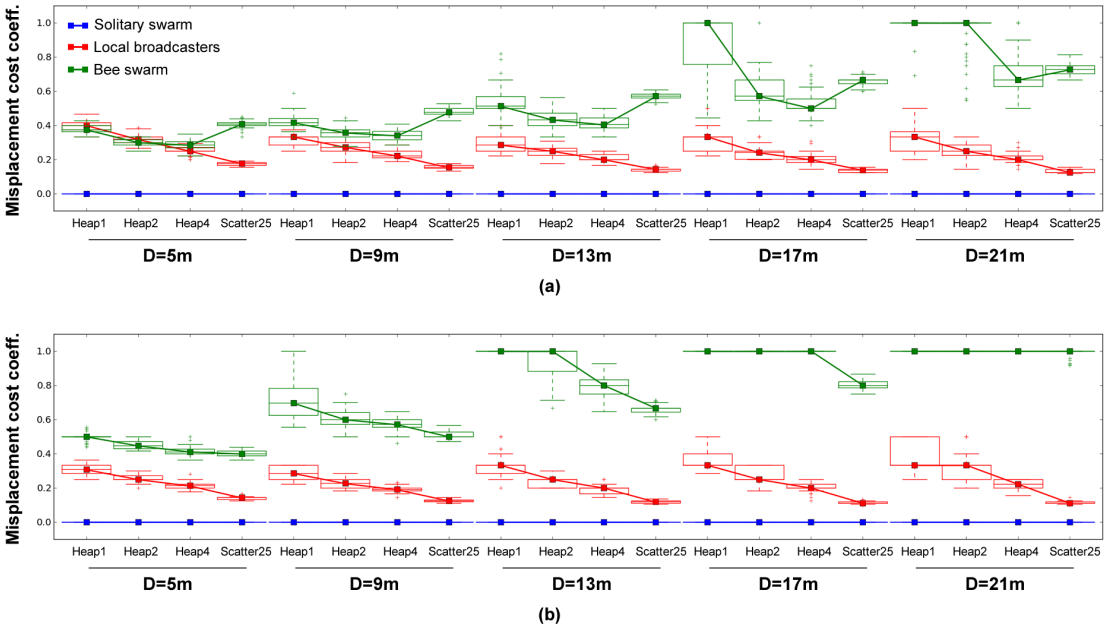


Figure 6.13: Misplacement cost coefficient, m, of 25-robot swarms in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was the highest in most scenarios and reached the maximum value of 1 in the most difficult scenarios, especially when the environmental change was fast.

worksite distance was large or when the environment changed quickly. In the most extreme cases, their $m = 1$, indicating that bee swarm robots were unable to use the information about worksites at all. This difficulty of bee swarms in the consumption task was caused by the fact that bee swarm scouts travelled to the base in order to recruit after they discovered a worksite. When the base was far away, or when worksite locations changed quickly, a worksite was very likely to disappear before a robot had a chance to return to it.

Consequently, bee swarms had the highest misplacement cost coefficient among all

swarms in the dynamic consumption task, especially when the environment changed quickly (see Figure 6.13 for results from 25-robot swarms and Appendix E, Figures E.23 and E.24 for results from 10- and 50- robot swarms, respectively). Their disadvantage was more pronounced in dynamic, compared to static environments (e.g., compare Figures 6.13 and 4.13).

6.1.4 Tendency to incur opportunity cost

Opportunity cost, C_O, (see Section 4.3.3) is incurred by robots that are subscribed to depleted worksites. These robots are missing an opportunity to explore the environment and to perform work. In the static consumption task, C_O can be incurred when a new recruit is unable to reach its worksite quickly enough. In the static collection task, a robot's worksite can also be depleted while the robot is unloading resource in the base or while it is traveling to the base. Additionally, in dynamic tasks, worksites are removed from the environment at the end of each change interval, which increases the probability that robots will incur C_O.

The tendency of swarms to incur opportunity cost in static environments was evaluated in Chapters 4 and 5 in terms of the *total* amount of C_O paid during a simulation run. In dynamic tasks, the *average* amount of C_O paid per change interval is reported instead, so that a meaningful comparison between static and dynamic environments can be made.

Solitary robots do not recruit, and their C_O thus equals to 0 in both the dynamic and the static consumption task. In the collection task, solitary swarms paid a higher amount of C_O per change interval in dynamic tasks, compared to the total amount of C_O paid in static task, in environments where it was likely that the robots would discover but not fully deplete worksites before the environment changed. For example, a higher C_O was paid when the environmental change was slow and when worksite distance from the base was small (see Figure 6.14 for results from 25-robot swarms and Appendix E, Figures E.25 and E.26 for results from 10- and 50- robot swarms, respectively). Most

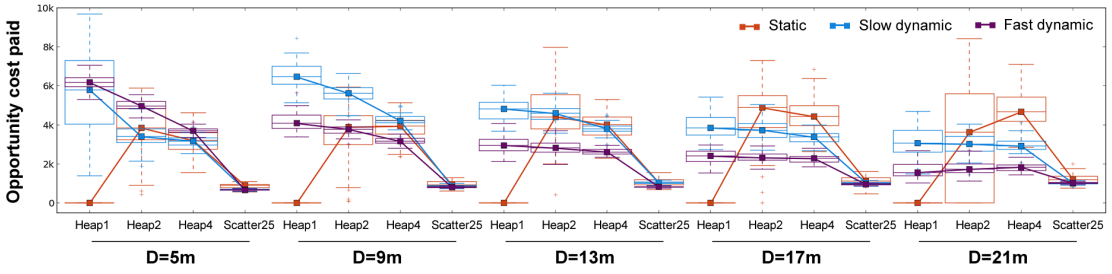


Figure 6.14: Opportunity cost, C_O, of 25-robot solitary swarms in the collection task. The swarms paid a higher C_O in dynamic, compared to static, environments when D was small. The amount of C_O paid in dynamic environments was usually higher when the environment changed slowly.

significantly, while the swarms did not pay any C_O in static Heap1 environments, since only a single worksite existed there, C_O was paid in dynamic Heap1 environments, where the worksite was repeatedly regenerated, and the amount of C_O paid was usually higher than that in other scenarios with the same D. On the other hand, a smaller amount of C_O was paid in dynamic, compared to static, environments when robots were less committed to worksites, most notably when the environment changed quickly and when worksites were far away from the base.

Similarly, local broadcasters paid a higher amount of opportunity cost when robots were likely to be subscribed to but not located at worksites at the end of a change interval. Because local broadcasters advertised worksite locations during the whole time while they were working, the strongest recruitment occurred in the slow consumption task with a few worksites that were close to the base. In these cases, the C_O paid by local broadcasters was the highest among all scenarios (see Figure 6.15 for results from 25-robot swarms and Appendix E, Figures E.27 and E.28 for results from 10- and 50- robot swarms, respectively).

Unlike solitary and local broadcaster swarms, bee swarms performing the dynamic consumption task usually paid the highest amount of C_O when environment changed quickly (see Figure 6.16a for results from 25-robot swarms and Appendix E, Figures E.29a and E.30a for results from 10- and 50- robot swarms, respectively), due to the fact that robots that discovered worksites were often unable to make a recruitment trip to the

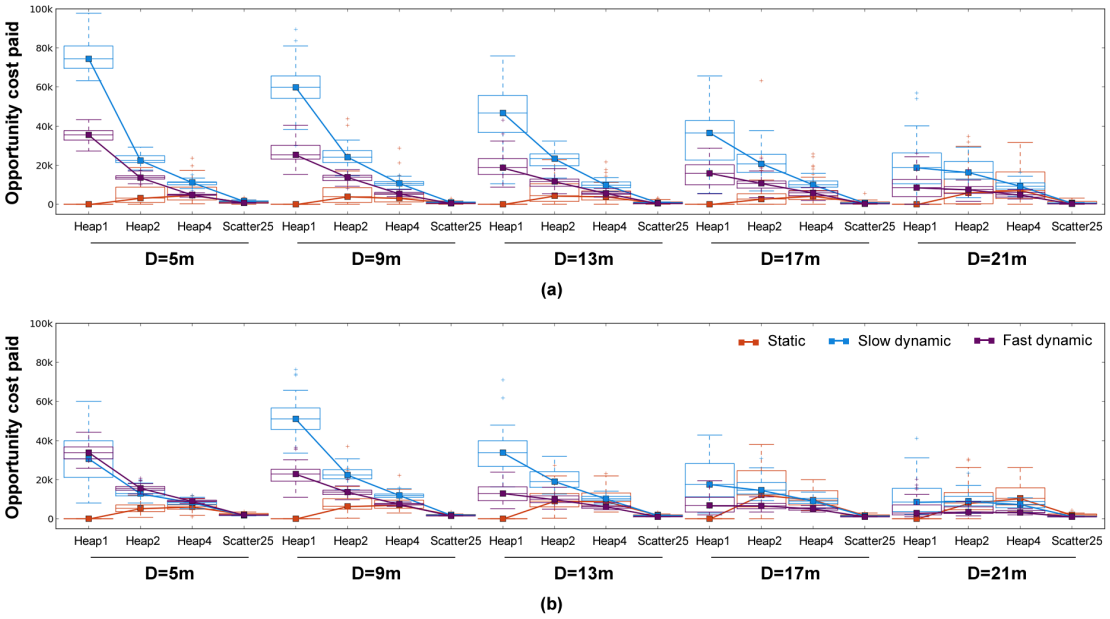


Figure 6.15: Opportunity cost, C_O, of 25-robot local broadcaster swarms in the (a) consumption and (b) collection task. The amount of C_O paid was usually the highest in slow dynamic environments, especially when a few worksites were located close to the base, and the lowest in static environments.

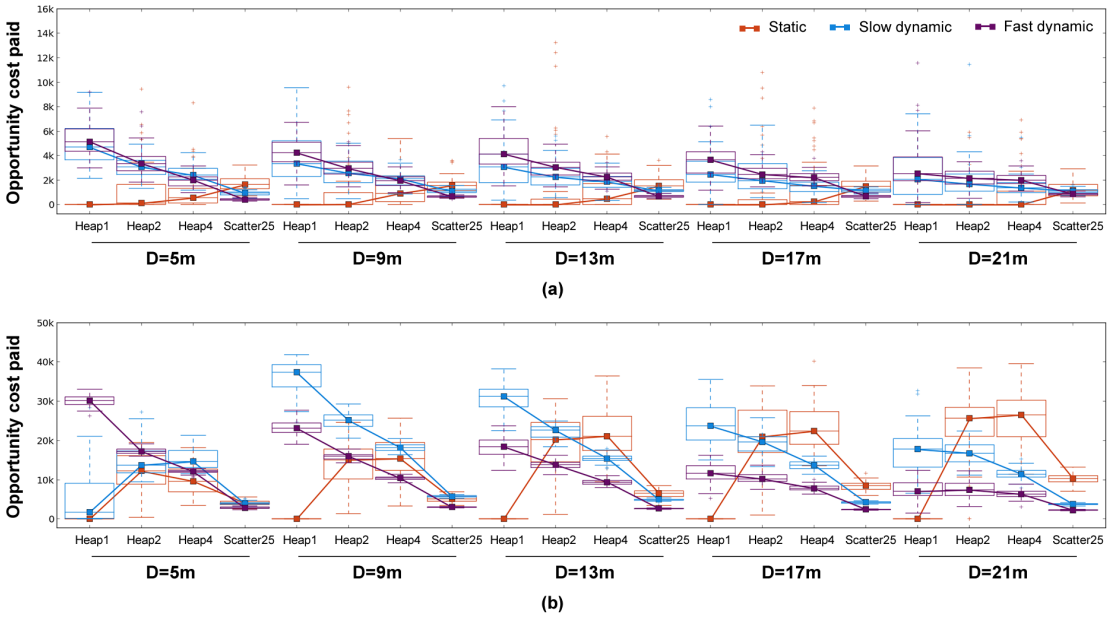


Figure 6.16: Opportunity cost, C_O, of 25-robot bee swarms in the (a) consumption and (b) collection task. In the consumption task, the amount of C_O paid was usually the largest in the dynamic environments and when the environment changed quickly. In the collection task, the swarms paid a larger C_O in dynamic, compared to static, environments when D was small. The amount of C_O paid was usually larger in slow, compared to fast, dynamic environments.

base and back before the environment changed. On the other hand, similarly as solitary and local broadcaster swarms, bee swarms paid a higher amount of C_O when the environment changed slowly, compared to when it changed quickly, in the collection task (Figure 6.16b and Appendix E, Figures E.29b and E.30b). The amount of C_O paid was higher compared to the static collection task when recruitment was strong, most significantly when the environment changed slowly and when worksites were close to the base.

As was the case in the static consumption task, local broadcasters paid the highest amount of C_O, compared to the other swarms, in both slow and fast dynamic consumption tasks in Heap environments, since they were very likely to find and recruit to worksites (see Figure 6.17 for results from 25-robot swarms and Appendix E, Figures E.31 and E.32 for results from 10- and 50- robot swarms, respectively). The amount of C_O paid by local broadcasters and bee swarms was similar and relatively low in Scatter25 environments, where a worksite could be depleted quickly and usually before a change interval ended. Solitary swarms did not pay any C_O.

In the dynamic collection task, bee swarms, where robots were recruited in the base and thus spent more time traveling to worksites, paid the highest amount of C_O in most environments, especially when the environmental change was fast (see Figure 6.18 for results from 25-robot swarms and Appendix E, Figures E.33 and E.34 for results from

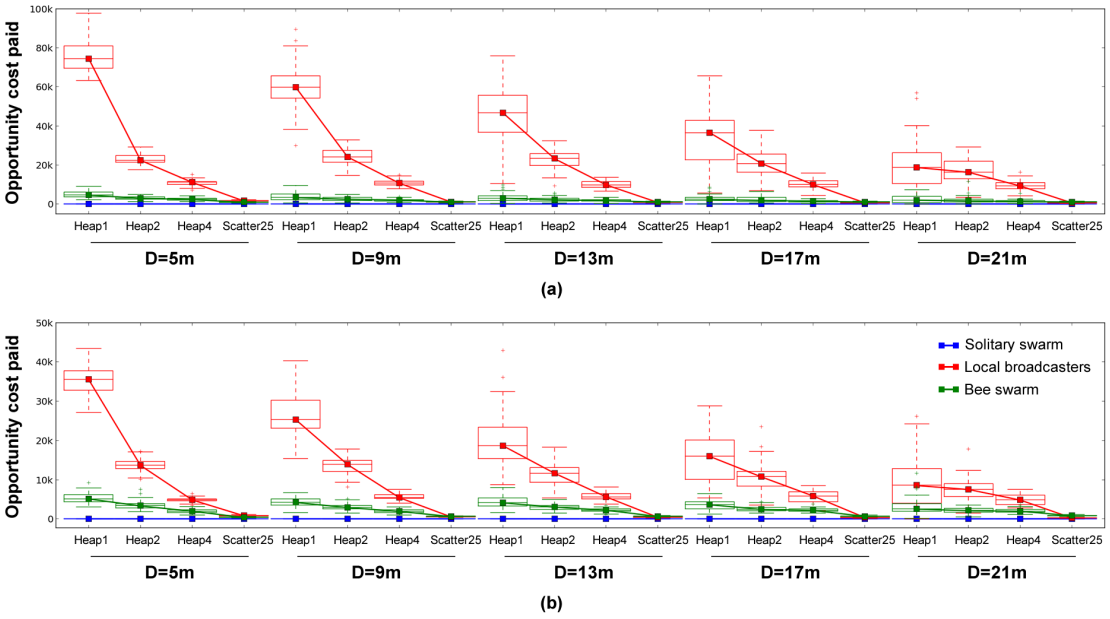


Figure 6.17: Opportunity cost, C_O, of 25-robot swarms in the (a) slow and (b) fast dynamic consumption task. Local broadcasters paid the largest C_O in Heap environments, followed by bee and solitary swarms.

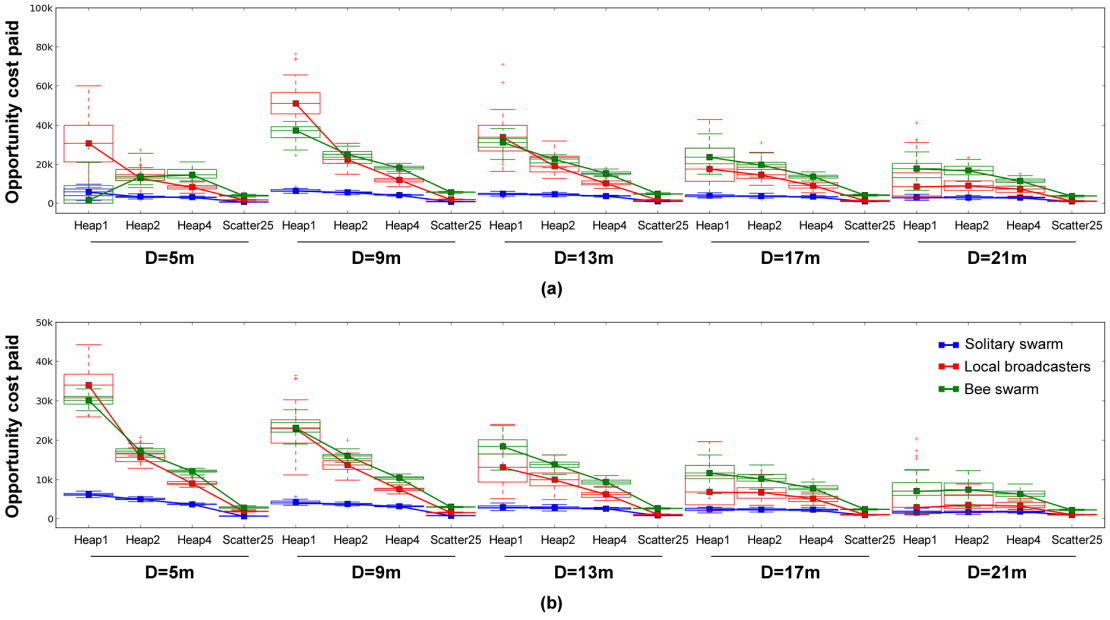


Figure 6.18: Opportunity cost, C_O, of 25-robot swarms in the (a) slow and (b) fast dynamic collection task. Local broadcasters paid the largest C_O in Heap1 environments when D was small. Bee swarms paid the largest C_O in most environments when D was large.

10- and 50- robot swarms, respectively). This result was similar to that from the static collection task, where bee swarms usually paid the highest amount of C_O, although the

differences between swarms were not as pronounced in the dynamic task. In contrast, local broadcasters paid the highest amount of C_O in a small number of cases, such as the Heap1 scenarios with a small D. Since they could discover and recruit to worksites faster than bee swarms in these scenarios, the number of robots subscribed to worksites that were removed at the end of a change interval was higher in local broadcaster, compared to in bee swarms.

6.2 Swarm-environment fit

Running experiments with the consumption and the collection task in dynamic environments offers a more complete insight into how the investigated robot control strategies interact with the environment. It is apparent that the four swarm characteristics, scouting efficiency, information gain rate and tendency to incur misplacement and opportunity costs, differ compared to when worksite locations are static.

Firstly, the scouting behaviour of robots has a more profound effect on swarm performance in dynamic environments. As was predicted by Hypothesis 6.1, bee swarms are not able to discover new worksites after the environment changes as quickly as the other swarms (see Section 6.1.1). Their difficulty is apparent regardless of worksite distance form the base, but it is stronger when worksites are further away. Since bee swarm scouts periodically return to the base in order to check whether other members of the swarm have any information to share, they are less likely than scouts of other swarms to be located near a newly added worksite. This also negatively affects the information gain rate, i, of bee swarms. Unlike when environments are static, bee swarms are unable to reach the highest i in many dynamic environments (see Section 6.1.2). This is especially the case in the dynamic consumption task, where local broadcasters recruit faster and more often, since, unlike bee robots, they do not travel to the base at all.

Figures 6.19 and 6.20 depict the winning strategies in the slow and fast dynamic consumption tasks and support the expectations of Hypothesis 6.2. Even though bee swarms perform similarly well compared to the other swarms in the static consumption task (see Figure 4.4), their performance is very rarely equivalent and never better that that of other swarms in the dynamic consumption task. While their poor performance is caused partially by their inefficient scouting, it is also low due to their inefficient use of information. Successful bee scouts travel to the base in order to recruit, causing the misplacement cost coefficient, m, of bee swarms to be often very high and to reach the maximum value of $m = 1$ in the most difficult dynamic environments (see Section 6.1.3). On the other hand, the m of solitary swarms and of local broadcasters is very often similar in static and dynamic consumption tasks.

In line with Hypothesis 6.3, solitary swarms, that are a winning strategy in only a small number of scenarios in the static consumption task (see Figure 4.4), perform better than

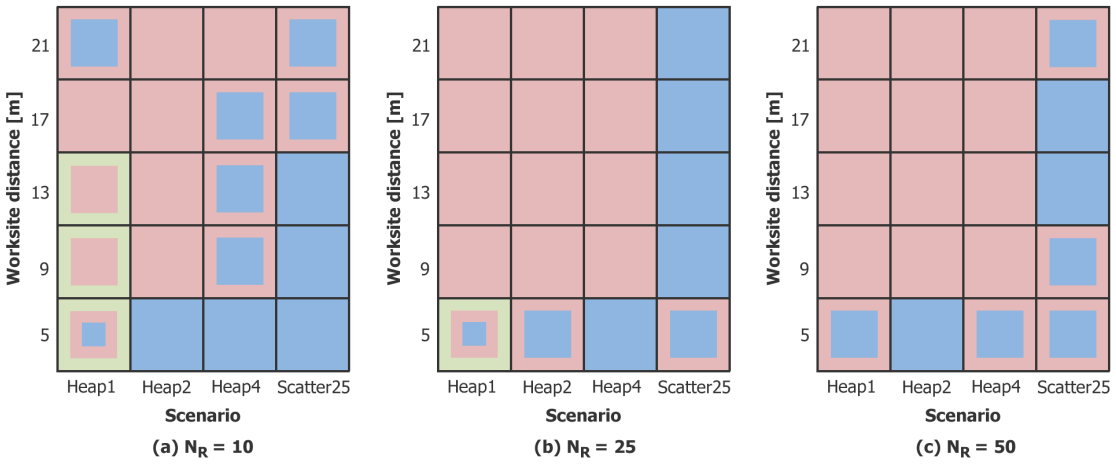


Figure 6.19: Winning strategies in the slow dynamic consumption task using (a) 10, (b) 25 and (c) 50 robots. The strategies are identified by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix E, Figures E.35a, E.36a and E.37a.

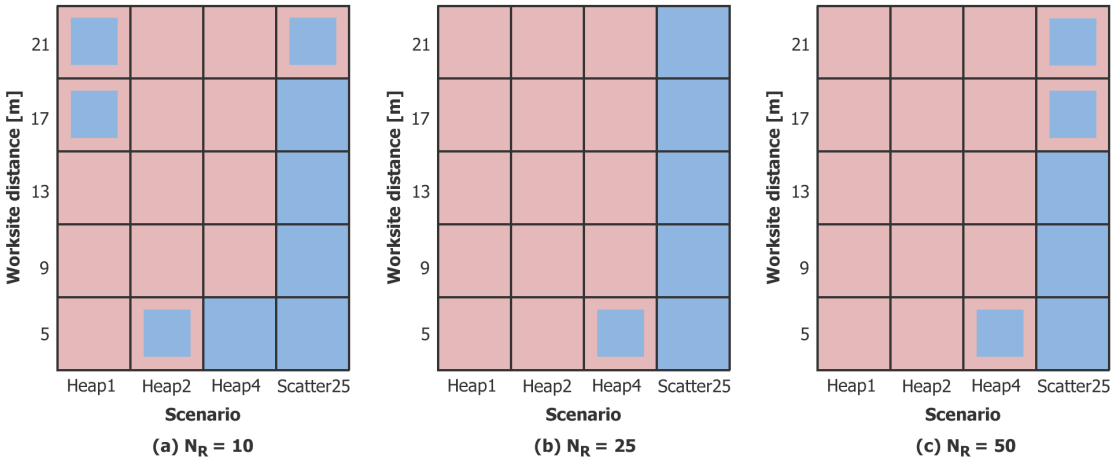


Figure 6.20: Winning strategies in the fast dynamic consumption task using (a) 10, (b) 25 and (c) 50 robots. The strategies are identified by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix E, Figures E.35b, E.36b and E.37b.

the other swarms in a much larger number of scenarios when the consumption task is dynamic. Solitary robots do not recruit and are thus better spread out in the work arena than robots from the other swarms, as it is indicated by the relatively low amount of C_O that they pay (see Section 6.1.4). This strategy is advantageous in dynamic environments when worksites are easy to discover, i.e., in the Scatter25 environments

and in the Heap environments with worksites very close to the base.

According to Hypothesis 6.3, local broadcasters, that pay the highest amount of C_O in Heap environments, should be outperformed by other swarms. This is however not true, and local broadcasters are a winning strategy in most Heap environments in the dynamic

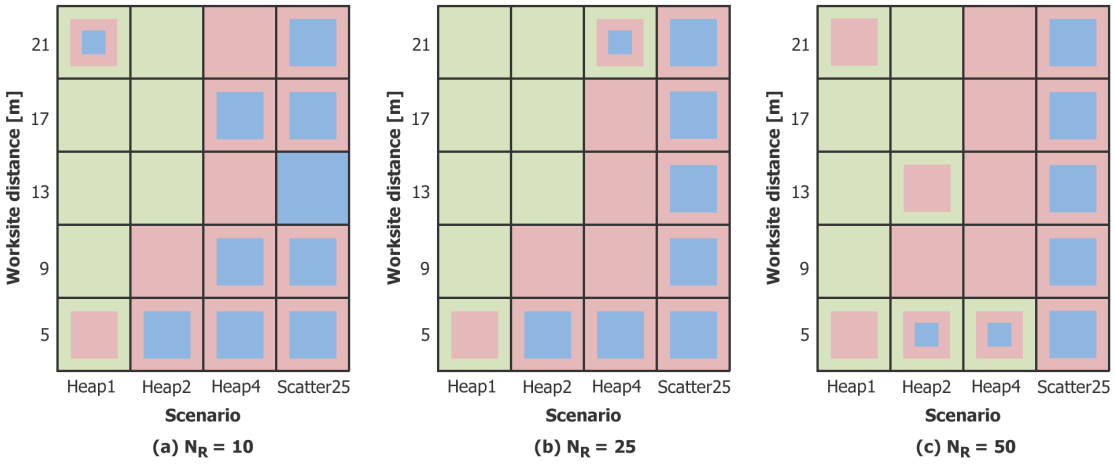


Figure 6.21: Winning strategies in the slow dynamic collection task using (a) 10, (b) 25 and (c) 50 robots. The strategies are identified by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix E, Figures E.38a, E.39a and E.40a.

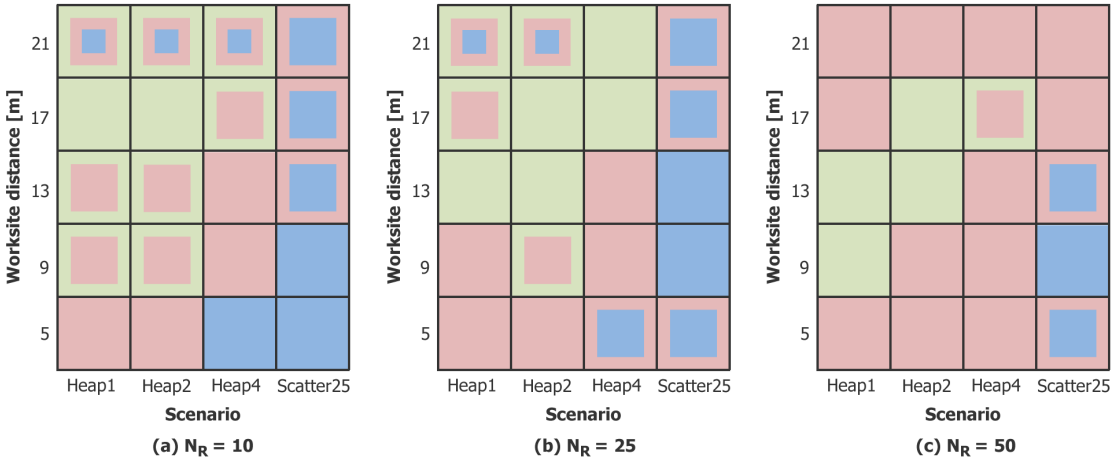


Figure 6.22: Winning strategies in the fast dynamic collection task using (a) 10, (b) 25 and (c) 50 robots. The strategies are identified by colour: solitary swarm (blue), local broadcasters (red), bee swarm (green). Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix E, Figures E.38b, E.39b and E.40b.

consumption task. There are two reasons for the unexpected success of this strategy. Firstly, solitary swarms cannot outperform local broadcasters in Heap environments, as worksites are difficult to discover and recruitment is thus necessary, since it increases a swarm's information gain rate. Secondly, bee swarms cannot outperform local broadcasters either, due to their ineffective exploration and exploitation of the environment. Even though local broadcasters do pay the highest amount of C_O in these cases, they trade it off for a relatively fast information gain rate and a relatively small amount of misplacement cost paid.

Winning strategies in the dynamic collection task are depicted in Figures 6.21 and 6.22. As was the case in the static collection task, bee swarms are the winning strategy in the most difficult environments, such as Heap1 and Heap2 with a large D, especially when the environment changes slowly (Figure 6.21). However, in line with Hypothesis 6.3, bee swarms do not perform as well as in the static collection task (see Figure 5.6) in scenarios with intermediate and low difficulty, such as the Heap2, Heap4 and Scatter25 scenarios with a small D. In these dynamic environments, bee swarm robots tend to overcommit to worksites and incur a high amount of C_O, which makes it difficult for them to respond to changes in the environment as quickly as solitary and local broadcaster robots. The disadvantage of bee swarms is more pronounced when the environment changes quickly (Figure 6.22).

At the same time, the advantage of solitary swarms over the other swarms is smaller in the dynamic collection, compared to the dynamic consumption task. Solitary swarms are the winning strategy in only a small number of scenarios and when the environment changes quickly. However, Hypothesis 6.3 is still supported in these cases, as solitary swarms never outperform the other swarms when the consumption task is static (see Figure 5.6). The difficulty of the solitary swarms is caused by their smaller information gain rate. In the collection task, robots travel back to the base to unload resource, meaning that they are not immediately available to sample the environment when worksite locations change. In such setting, being able to share information among robots becomes more important, causing strategies that do utilise recruitment to outperform solitary swarms.

6.3 Summary and discussion

In a dynamic environment, worksite locations change periodically and robots are thus required to continuously scout the environment and to extract reward from worksites as quickly as possible. The scouting strategy of swarms, as well as their tendency to incur misplacement and opportunity costs, thus have a more profound effect on the swarm performance, compared to when the environment is static.

Solitary swarms, where robots do not recruit and are thus evenly spread out in the work arena, outperform the other swarms in more scenarios when environments are dynamic, compared to when they are static. This is especially true in the consumption task, where robots do not need to travel to the base to obtain reward. Furthermore, solitary swarms enjoy a greater advantage in both the consumption and the collection tasks when the environment changes quickly, i.e., when it is most important to discover new worksites as soon as possible. However, as is the case in static environments, solitary swarms are unable to outperform swarms that utilise recruitment in scenarios where worksites are difficult to discover.

Bee swarm robots, that scout less efficiently than robots from the other swarms, and that also use information less efficiently, since they are usually misplaced from reward for long periods of time, are less suitable in dynamic tasks, unless they are performing the dynamic collection task in scenarios where worksites are very difficult to discover. Local broadcasters outperform bee swarms in a larger number of scenarios with intermediate difficulty, compared to when tasks are static. This is especially true for the dynamic consumption task, where local broadcasters collect more resource than the solitary and the bee swarms in majority of the explored scenarios.

Similar dynamic tasks have been explored in order to model various swarm applications, such as servicing of machines in a warehouse (Stiefelhagen et al., 2004), picking up and delivering of goods (Thiel et al., 2009; Wawerla and Vaughan, 2010) or solving a general problem of "task allocation" in multi-robot systems (Gerkey and Mataric, 2003; Mataric et al., 2003; Sarker and Dahl, 2011). Various approaches have been taken, including using a central planner that collects data from robots and instructs them what to do (Stiefelhagen et al., 2004; Wawerla and Vaughan, 2010), auction-based approaches, where robots bid for worksites through a centralised server (Mataric et al., 2003; Thiel et al., 2009), as well as various decentralised approaches where robots communicate worksite locations to other robots nearby and recruit them (Gerkey and Mataric, 2003; Wawerla and Vaughan, 2010; Sarker and Dahl, 2011). However, to best of the author's knowledge, multiple control strategies have not yet been compared in a single set of experiments to the same extent as has been carried out here. It is therefore not clear from the literature what types of algorithms are suitable in various dynamic tasks and for what reasons.

The comparative studies presented in Chapters 4 – 6 are a step towards establishing a set of principles that can guide design of robot control algorithms. Before the lessons learned from these experiments are formalised as design patterns in Chapter 9, two biologically inspired add-on control strategies are investigated, in order to gain further insight into how dynamic tasks can be handled by swarms. The add-on strategies are applied to each of the three swarms (solitary, local broadcaster and bee swarms) in conjunction with their basic behaviour. The aim of these experiments is to find out whether they help the swarms to overcome particular difficulties presented by dynamic environments, such as

obtaining reward as quickly as possible, or making sure that robots do not overcommit to worksites.

Chapter 7

The Opportunism add-on strategy

It was argued in Chapter 6 that it is important for swarms to be able to extract reward from worksites as quickly as possible when an environment is dynamic, i.e., when worksites are only available for a certain period of time. Additionally, especially in the Scatter25 scenarios, robots need to spread out evenly across the work arena in order to visit as many worksites as possible and avoid paying misplacement and opportunity costs due to physical and exploitational interference.

In this chapter, a way of helping the swarms to extract more reward from the environment is explored. All three swarms, solitary, local broadcaster and bee swarms, are equipped with an add-on control strategy, *Opportunism*. With Opportunism, robots preferentially exploit worksites that have higher returns, i.e., those that have larger volumes. Additionally, in the collection task, worksites that are closer to the base are preferred. It is expected that:

Hypothesis 7.1. Opportunism improves the performance of a control strategy in dynamic environments, provided that robots are able to consider information about a number of worksites at the same time.

In order for the robots to be able to compare the worksites that they encounter, each robot measures the *utility*, U_W, of a worksite that it is close by. In the *consumption task*, the U_W of a worksite is simply equal to its current volume, V_W:

$$U_W = V_W \tag{7.1}$$

In the *collection* task, the distance, D_W, between a worksite and the base edge is also considered, so that worksites that are further away from the base have a lower utility, as it takes longer to receive reward from them[1]:

$$U_W = V_W / D_W \tag{7.2}$$

Bee swarm robots, that return to the base in both the collection and the consumption task (see Section 3.1.4), always calculate U_W based on Equation 7.2.

The three basic control strategies are expanded by Opportunism as follows:

- A robot measures utility, U_W, of a worksite that it performs work at. In the consumption task, U_W is measured each second while a robot is working. In the collection task, U_W is measured before a robot leaves the worksite to unload resource in the base.
- Robots with all control strategies additionally evaluate the utility, U_N, of any *new* worksite that they encounter. In the consumption task, recruits evaluate new worksites while traveling to the advertised location. In the collection task, new worksites are additionally evaluated if a robot encounters them on its round trip to the base.
- Robots that utilise recruitment exchange information about utility of the worksites that they are currently working on during the parts of their work cycle when they normally communicate. Local broadcasters do so while around worksites and bee swarm robots do so in the base.
- A robot abandons its current worksite and switches to a new one when $E_N > E_W$.

Opportunism affects the behaviour of the basic control strategies in the following ways:

- Solitary robots are only able to use Opportunism during the collection task, when they encounter other worksites on their way to the base and back.
- Since local broadcaster and bee swarm robots can be recruited, they evaluate the utility of new worksites encountered on their way to the advertised worksite location. They can thus apply Opportunism in both the consumption and the collection tasks.
- Local broadcasters additionally evaluate the worksite utilities broadcasted by all working robots nearby. It is thus possible that a local broadcaster scout or worker switches to a new worksite while it is subscribed to another one. For example, a scout may receive recruitment signals from multiple directions and will always follow to a worksite with the highest advertised utility. Furthermore, if the distance between worksites is smaller than the communication range of robots, the

[1]The measure of utility in the collection task has been inspired by food patch *energy efficiency* used by honey bees to evaluate profitability of a foraging site (Seeley, 1994).

robots exchange the information about the utilities of their worksites each second while they are working. If worksite A is being depleted quicker than worksite B (for example, because more robots are working at A), its utility decreases faster, meaning that robots from A will gradually be recruited to B.

- Bee swarm robots exchange information about the utility of their worksites while they recruit in the base. If there are multiple worksites being advertised at the same time, all recruiters gradually adopt the worksite with the highest advertised utility. Additionally, new recruits that are leaving the base and encounter a signal about a better worksite subscribe to it as well.

In the following sections, the impact of Opportunism on swarm characteristics is evaluated, followed by an account of when this add-on control strategy improves the performance of swarms. It is shown that the behaviour of local broadcasters is in line with Hypothesis 7.1 in some of the explored environments. On the other hand, bee swarms perform worse when the behaviour is applied, due to the fact that they exchange information in a single designated place, the base, which causes large groups of robots to concentrate on a single worksite. Solitary robots are mostly not affected, since they are rarely in a situation when they can use Opportunism.

7.1 Swarm characteristics

7.1.1 Scouting efficiency

The scouting efficiency (see Section 6.1.1) of solitary swarms and of local broadcasters was not affected by Opportunism and remained similar for any task, swarm size, scenario or worksite distance.

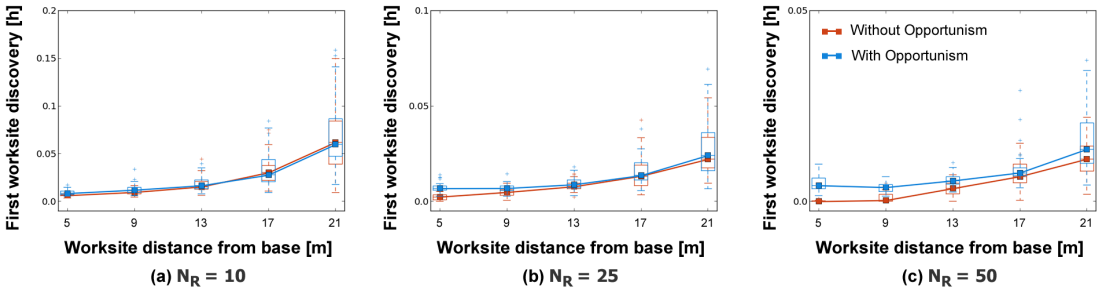


Figure 7.1: Time of the first worksite discovery, in the slow dynamic consumption task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 bee swarm robots with and without Opportunism. Bee swarms of 50 robots could not discover their first worksite as quickly with Opportunism.

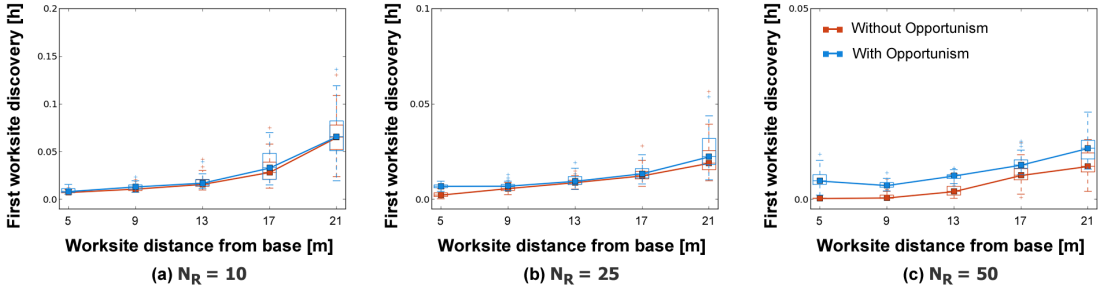


Figure 7.2: Time of the first worksite discovery, in the fast dynamic consumption task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 bee swarm robots with and without Opportunism. Bee swarms of 50 robots could not discover their first worksite as quickly with Opportunism.

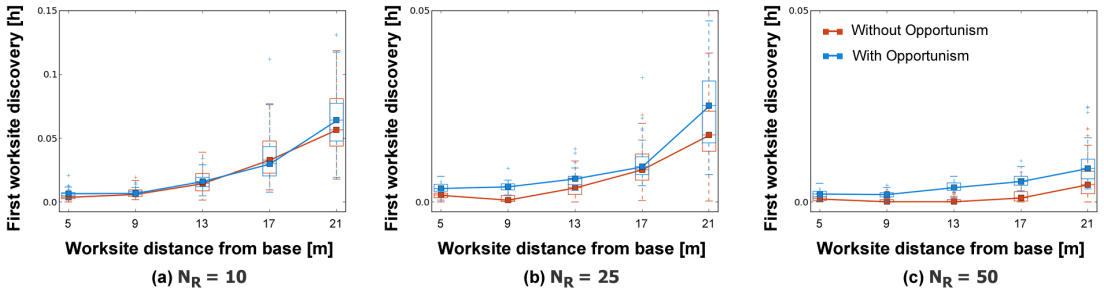


Figure 7.3: Time of the first worksite discovery, in the slow dynamic collection task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 bee swarm robots with and without Opportunism. Bee swarms of 50, and in some cases of 25, robots could not discover their first worksite as quickly with Opportunism.

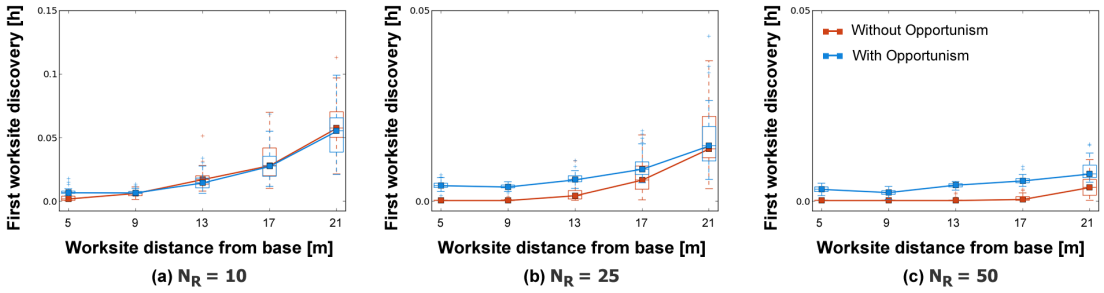


Figure 7.4: Time of the first worksite discovery, in the fast dynamic collection task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 bee swarm robots with and without Opportunism. Bee swarms of 50, and in some cases of 25, robots could not discover their first worksite as quickly with Opportunism.

The scouting efficiency of bee swarms was mostly not affected by Opportunism when the swarms were small ($N_R = 10$) or when there were a few worksites in the environment. However, in Scatter25 scenarios, large ($N_R = 50$), and in some occasions medium-sized ($N_R = 25$), bee swarms took longer to discover the first worksite with Opportunism (see Figures 7.1 – 7.4). The decrease in scouting efficiency was apparent in both the dynamic collection and consumption tasks. Opportunism, when used by bee swarms, caused large groups of robots to concentrate on a single worksite. This was especially problematic when the swarm was large, since more robots were available to carry and transmit information about the worksite with the highest utility. As a result, robots got stuck in congestion around the advertised worksite, which prevented them from scouting the environment.

7.1.2 Information gain rate

Like scouting efficiency, the information gain rate, i, (see Section 4.2.3) of solitary swarms and of local broadcasters was not affected by Opportunism.

The i of bee swarms was higher with, compared to without, Opportunism when strong recruitment (see Section 7.1.1) occurred in the dynamic collection task, i.e., when worksites were close to the base in the Scatter25 scenario and when swarms of 25 or 50 robots were used (see Figure 7.5 for results from 25-robot swarms and Appendix F, Figures F.1

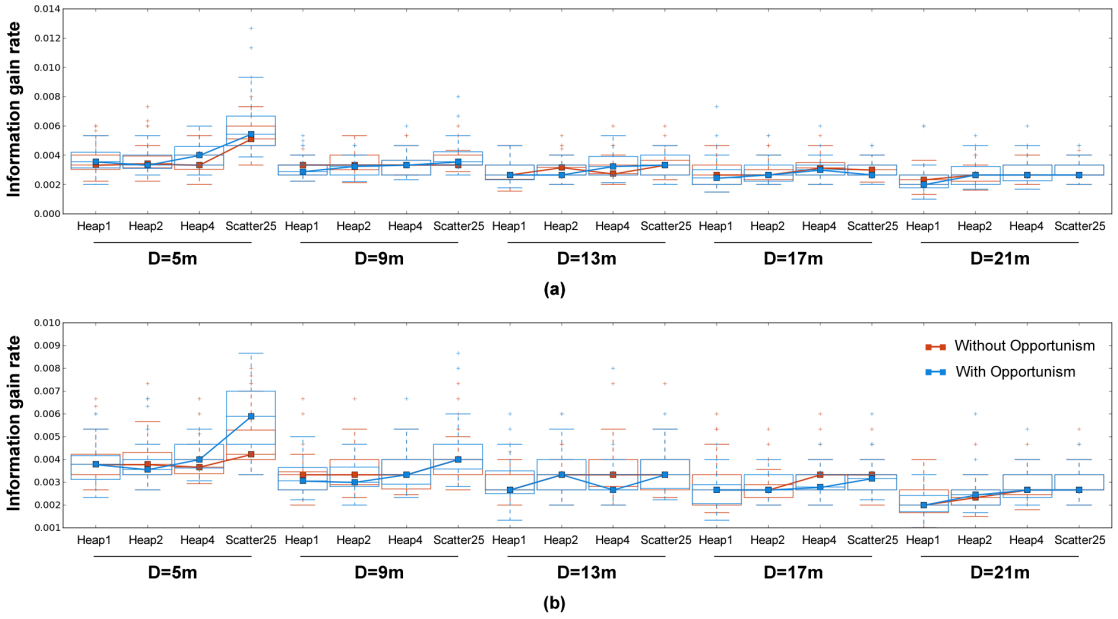


Figure 7.5: Information gain rate, i, of 25-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The i of bee swarms was higher with Opportunism when the environment changed quickly in the Scatter25 scenarios with $D = 5$m.

and F.2 for results from 10- and 50- robot swarms, respectively). The information gain rate did not increase in scenarios where worksite discovery and recruitment were less probable, i.e, where worksites were further away from the base, or in the case of 25-robot swarms, when the environment changed slowly. Similarly, the i of bee swarms with Opportunism was not higher in the consumption task, since robots travelled back to the base less often and thus the information about newly discovered worksites did not spread as quickly.

7.1.3 Tendency to incur misplacement cost

The misplacement cost, C_M, and the misplacement cost coefficient, m, (see Section 4.3.2) of solitary swarms were not affected by Opportunism. Since solitary robots did not recruit, their $m = 0$ in the consumption task. In the collection task, where the robots traveled to the base to unload resource, the median m of solitary swarms was always equal to 1. Similarly, the median m of local broadcasters and bee swarms was equal to 1 in the collection task.

In the consumption task, the m of local broadcasters that used Opportunism was lower than that of the basic local broadcaster swarms when worksites were close to the base (see Figure 7.6 for results from 25-robot swarms and Appendix F, Figures F.3 and F.4 for results from 10- and 50- robot swarms, respectively). This indicates that a smaller portion of robots that knew about worksites incurred misplacement cost. In Scatter25, where multiple worksites were often within the communication range of robots, new recruits sometimes switched to worksites that they found on their way to the originally advertised location. This caused them to stop incurring C_M sooner than scouts that did not use Opportunism and that always arrived to the advertised worksite.

Perhaps more surprisingly, the m of local broadcasters that used Opportunism was also smaller in Heap environments, despite the fact that worksites were far away from each other and it was thus not possible for a recruit to discover a new worksite on its way to the advertised location. This result can be explained by looking at how local broadcasters communicate. Due to random wheel sensor noise, a communicated worksite location could sometimes be inaccurate, especially in the Heap environments with a small D, where there was usually congestion near worksites that caused the robots to make numerous turns and their odometry-based vector to be incorrect. These errors rarely propagated through the swarm when Opportunism was not used, since robots exchanged information about a worksite only once, when a scout met with a worker. However, when Opportunism was used, local broadcasters exchanged information every second, making the possibility of error propagation much larger[2]. Robots that received an incorrect worksite location travelled to an empty space in the work arena, performed neighbourhood search and abandoned the worksite upon not being able to discover it, at which point they stopped incurring C_M. On the other hand, when Opportunism was

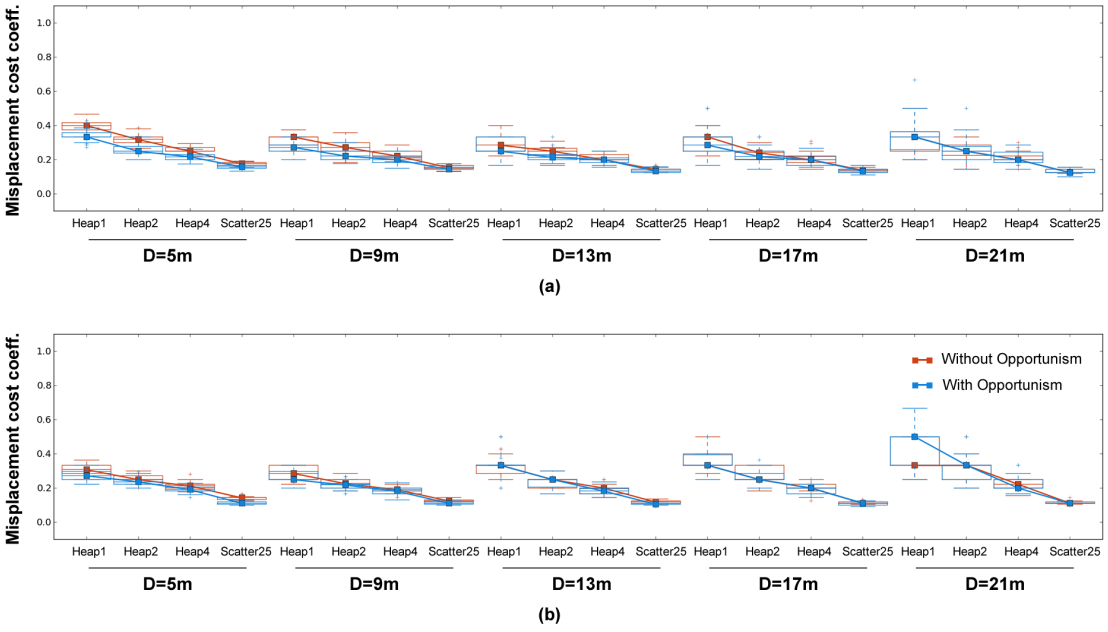


Figure 7.6: Misplacement cost coefficient, m, of 25-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The m of local broadcasters was smaller with Opportunism when D was small, especially when the environment changed slowly.

not used, a new recruit kept attempting to reach the advertised and congested worksite, which caused it to incur C_M for a much longer period of time.

Contrary to local broadcasters, bee swarms performing the consumption task paid a higher misplacement cost in easy environments, such as Scatter25 and Heap4, especially when worksites were close to the base (see Figure 7.7 for results from 25-robot swarms and Appendix F, Figures F.5 and F.6 for results from 10- and 50- robot swarms, respectively). Generally, there are two reasons why bee swarms can have a higher m: the proportion of recruited robots versus scouts is higher, or there is more congestion that prevents robots from reaching their worksites. Since the information gain rate of bee swarms was similar in these environments regardless of whether Opportunism was used (see Section 7.1.2), it is reasonable to assume that Opportunism caused a higher congestion in bee swarms. As a result of strong recruitment to a single worksite (see Section 7.1.1), congestion often prevented robots reaching their worksites and caused

[2]A more complicated algorithm, that for example prevents local broadcasters that use Opportunism from being re-recruited to the same worksite can prevent such errors, but only to a certain extent. In the simulation discussed here, local broadcasters could not be re-recruited to the same worksite for 30 seconds of "cool-off" time. During 30 seconds, robots that did not experience congestion could travel far enough from a worksite to be out of the communication range of any robots working on it. However, the length of the cool-off time could not be set to an arbitrarily large number, as re-recruitment to the same worksite was sometimes desirable. This was the case especially especially in Scatter25 scenarios, for example when another worksite, while originally better, became almost depleted, while the utility of a robot's original worksite remained similar. While the cool-off time decreased the frequency of re-recruitment, it could not completely eliminate it, especially in Heap scenarios, where congestion prevented the robots from getting out of the communication range of each other.

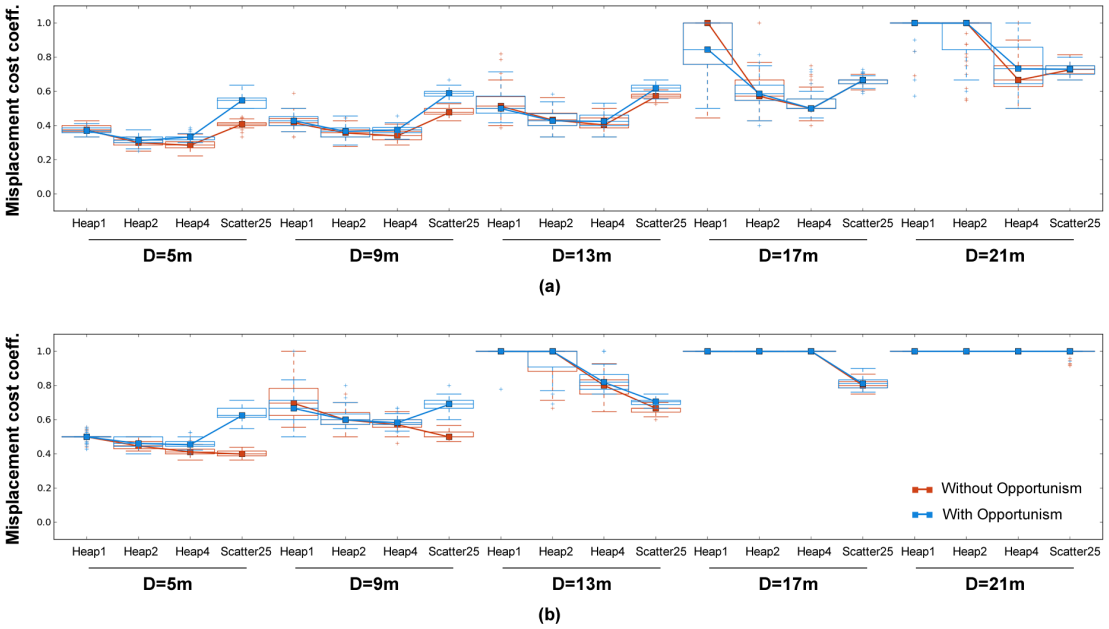


Figure 7.7: Misplacement cost coefficient, m, of 25-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was larger with Opportunism in Heap4 and Scatter25 scenarios when D was small.

them to incur a high amount of C_M. The negative effect of Opportunism was especially apparent when worksites were numerous, i.e., when robots were required to spread out in the work arena, instead of congregating around a single worksite.

7.1.4 Tendency to incur opportunity cost

Similarly as other swarm characteristics, the amount of opportunity cost, C_O, (see Section 4.3.3) paid by solitary swarms was not affected by Opportunism.

Local broadcasters paid a smaller amount of C_O when they used Opportunism, especially in the dynamic consumption task in Heap environments (see Figures 7.8 and 7.9 for results from 25-robot swarms and Appendix F, Figures F.7 – F.10 for results from 10- and 50- robot swarms, respectively). Like misplacement cost, opportunity cost of local broadcasters in these environments was a result of congestion near worksites that prevented recruits from arriving to the advertised worksite location and discovering that the worksite was depleted by other robots or removed as a result of the dynamics of the environment (i.e., at the end of a T_C interval, see Chapter 6). The Opportunism-induced communication errors (see Section 7.1.3) caused some robots to travel to an incorrect place and to abandon the worksite earlier than when Opportunism was not used. On the other hand, the opportunity cost paid by local broadcasters with and

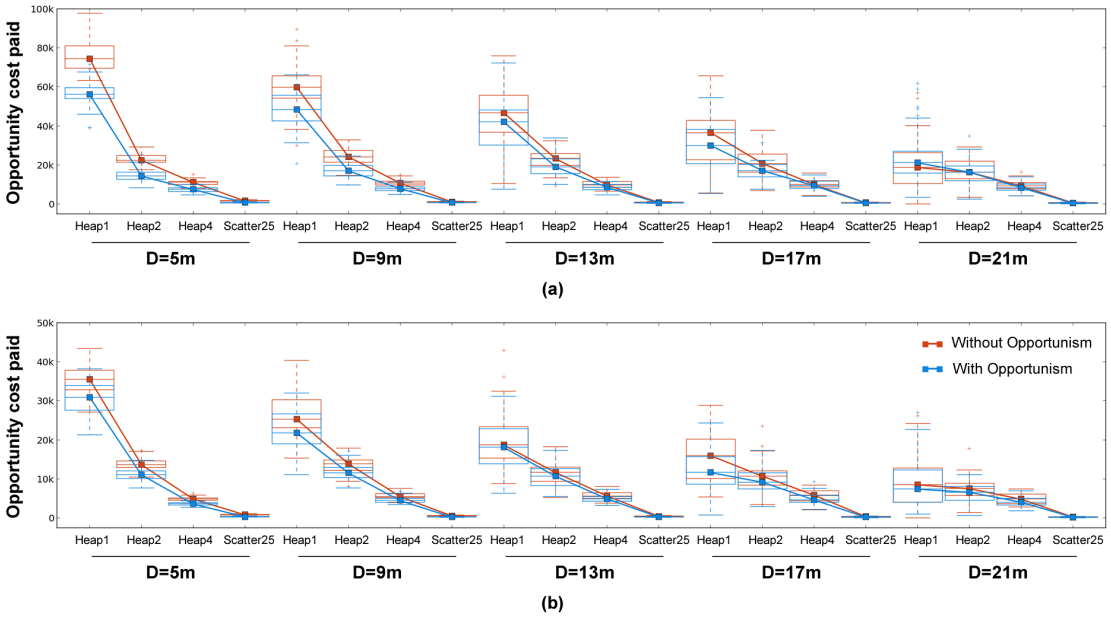


Figure 7.8: Opportunity cost, C_O, of 25-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by local broadcasters was smaller with Opportunism in Heap scenarios, especially when D was small and when the environment changed slowly.

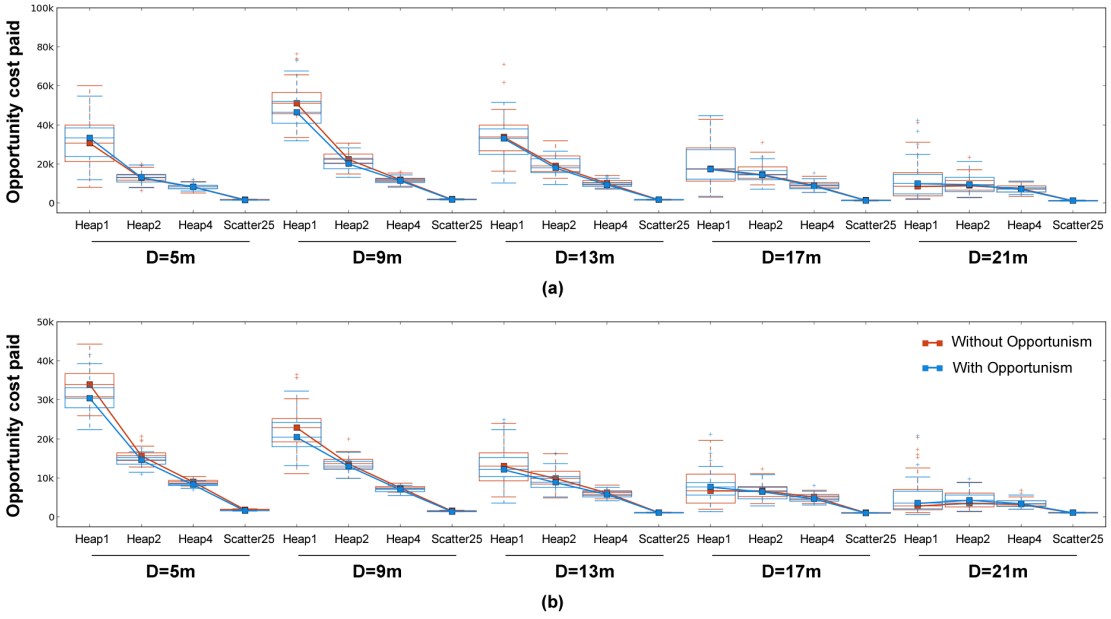


Figure 7.9: Opportunity cost, C_O, of 25-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by local broadcasters was smaller with Opportunism in some Heap1 scenarios.

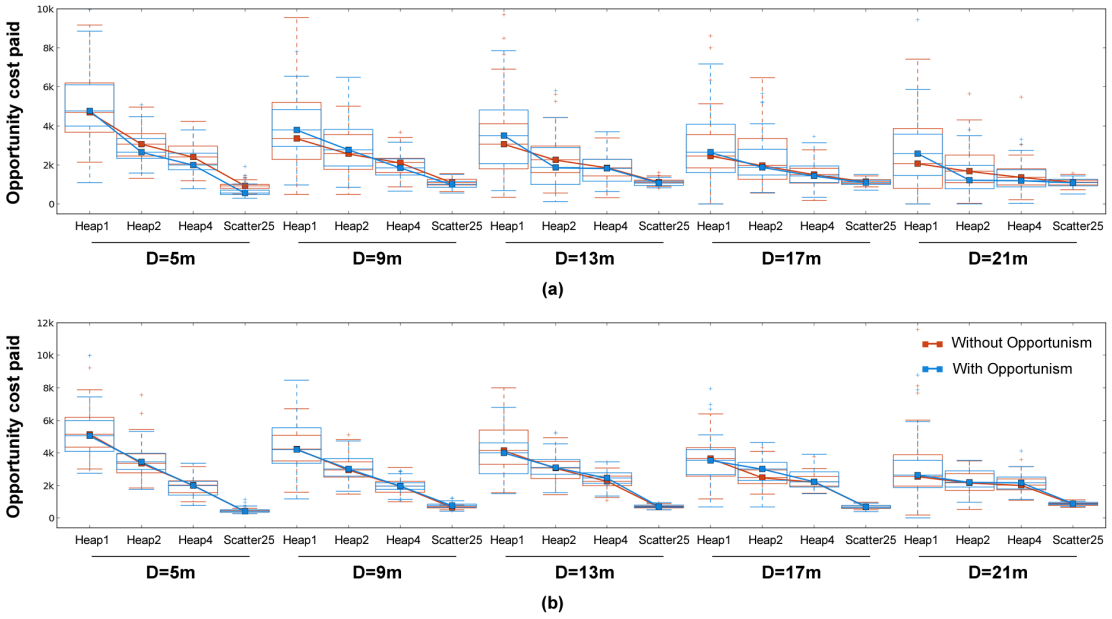


Figure 7.10: Opportunity cost, C_O, of 25-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The C_O paid by bee swarms was not affected by Opportunism in most of the scenarios.

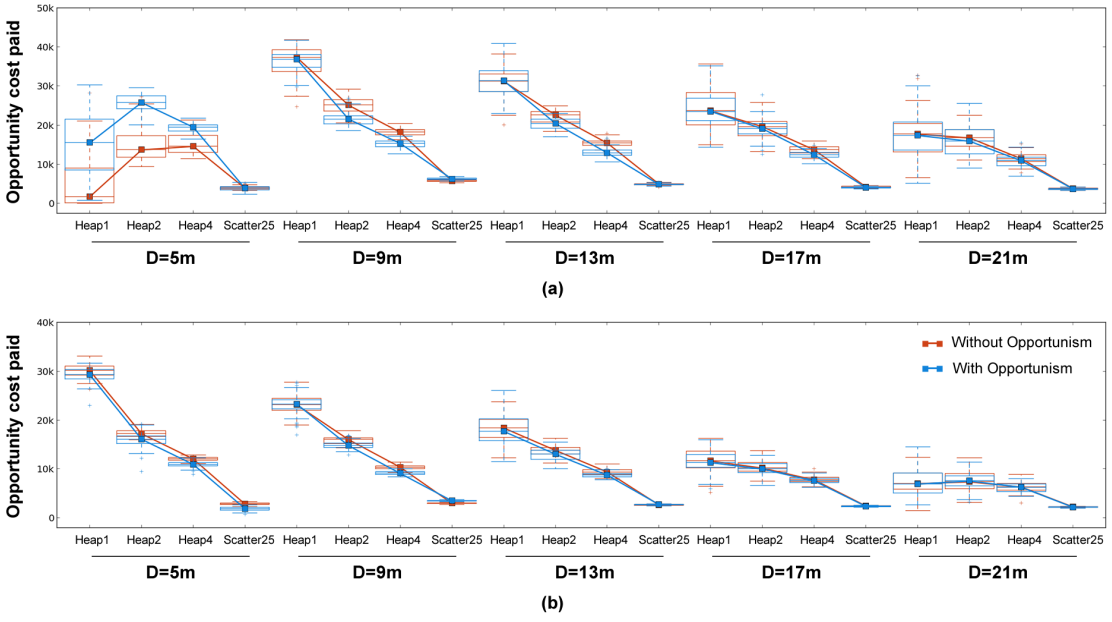


Figure 7.11: Opportunity cost, C_O, of 25-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by bee swarms was larger with Opportunism in Heap scenarios when $D = 5$m and when the environment changed slowly. The amount of C_O paid was smaller with Opportunism in some Heap2 and Heap4 scenarios when $D > 5$m, especially when the environment changed slowly.

without Opportunism was similar in Scatter25 scenarios, where worksites had small volumes and where congestion around them occurred less often and was less severe.

The opportunity cost of bee swarms was lower with, compared to without, Opportunism in a number of scenarios in the dynamic collection and consumption tasks, although the decrease was usually fairly small (see Figures 7.10 and 7.11 for results from 25-robot swarms and Appendix F, Figures F.11 – F.14 for results from 10- and 50- robot swarms, respectively). In these cases, Opportunism helped the robots with outdated information, that were unloading resource or recruiting in the base, to adopt new worksites, usually those that did not have many robots subscribed to them or those that were newly added to the environment at the end of a T_C interval, and thus had a higher utility, i.e., a larger volume. More significantly, Opportunism increased the amount of C_O incurred by bee swarm robots when strong recruitment was likely to cause severe congestion around a single worksite. This was the case in the collection task when worksites were very close to the base ($D = 5$m) and when the environment changed slowly.

7.2 Swarm performance

It is apparent from the analysis of swarm characteristics that the impact of Opportunism on the swarms depends on the basic robot control strategy used, as well as on the structure of the environment and the type of the swarm's task.

Solitary swarms can only utilise Opportunism in Scatter25 scenarios during the collection task, i.e., when they travel between worksites and the base and where it is possible for worksites to be located between the base and a robot's original work location. However, even in these cases, Opportunism is triggered very rarely. While there is a statistically significant effect of the add-on strategy on the performance of solitary swarms in some Scatter25 scenarios (see Figure 7.12 for results from 25-robot swarms and Appendix F, Figures F.15 and F.16 for results from 10- and 50- robot swarms, respectively), the difference in the amount of resource collected with Opportunism is usually very small (see Appendix F, Figures F.21 – F.23).

Figure 7.13 shows the effect of Opportunism on the performance of 25-robot local broadcasters swarms. Results for 10- and 50-robot swarms are shown in Appendix F, Figures F.17 and F.18. Opportunism has three main effects on the behaviour of local broadcasters:

1. In line with Hypothesis 7.1, recruited robots are able to find worksites with larger volumes, i.e. a larger utility, on their way to the advertised location in Scatter25 scenarios, where it is possible for multiple worksites to be located within the communication range of robots. This allows local broadcasters to spread out across multiple worksites better, increasing their performance, especially when the

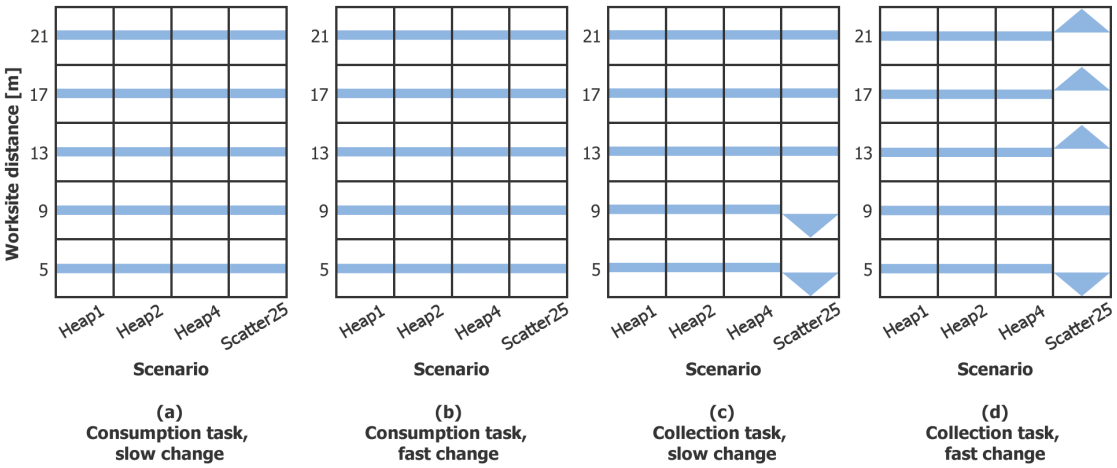


Figure 7.12: The effect of Opportunism on the performance of 25-robot solitary swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix F, Figure F.22.

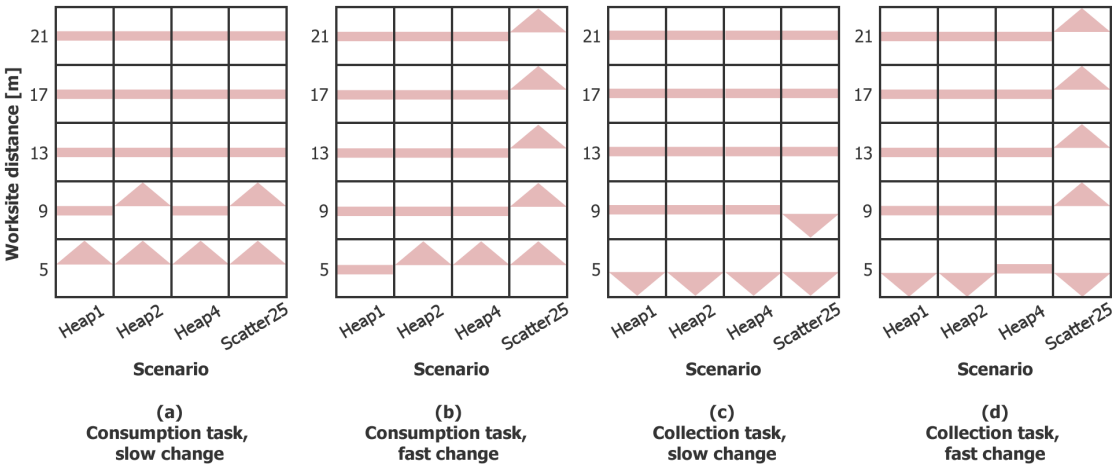


Figure 7.13: The effect of Opportunism on the performance of 25-robot local broadcaster swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix F, Figure F.25.

environment changes quickly, i.e., when it is more important to exploit as many worksites as possible. The performance improvement is stronger when worksites are closer to the base, i.e., when subscribed robots are more likely to encounter multiple worksites.

2. In Heap scenarios with a small D, recruits often cannot reach an advertised worksite due to congestion around it. Robots in congested areas often make a lot of turns while avoiding each other, which causes their odometry-based vector to the worksite to become increasingly incorrect. With Opportunism turned on, robots communicate about the worksite utility and location periodically, meaning that some recruits are sent to the wrong location. Rather than taking part in the congestion, these robots travel to the wrong place and eventually abandon the worksite and resume scouting. Not only they stop incurring the misplacement and the opportunity cost sooner than when Opportunism is not used, they are also able to discover new, non-congested, worksites. This is advantageous in the consumption task, where robots normally remain at worksites until they are depleted and where congestion is only cleared when a worksite is removed from the environment. The performance of local broadcasters with Opportunism is thus better than when Opportunism is not used in the consumption task when worksites are close to the base (Figure 7.13a,b).

3. On the other hand, in the collection task, where robots travel to the base to unload resource and where congestion around worksites is thus cleared periodically, the Opportunism-induced odometry and communication errors decrease the performance of local broadcasters in Heap scenarios with a small D (Figure 7.13c,d). In these cases, it is more advantageous for robots to wait until the path to their worksite clears, rather than to travel away and search for new worksites.

The effect of Opportunism on bee swarm performance is shown in Figure 7.14 and in Appendix F, Figures F.19 and F.20. Contrary to the prediction of Hypothesis 7.1, bee swarms that use Opportunism spread themselves less evenly in the environment, compared to when the add-on strategy is not used. Since bee swarm robots exchange information at a designated location in the base, the preference for a single worksite that has the highest utility often spreads to the majority of robots. This increases the amount of congestion around that worksite and increases the misplacement and the opportunity cost incurred by the robots. The performance of bee swarms is thus lower with, compared to without, Opportunism in many scenarios, especially in the collection task, where robots exchange information every time they return to the base to unload resource.

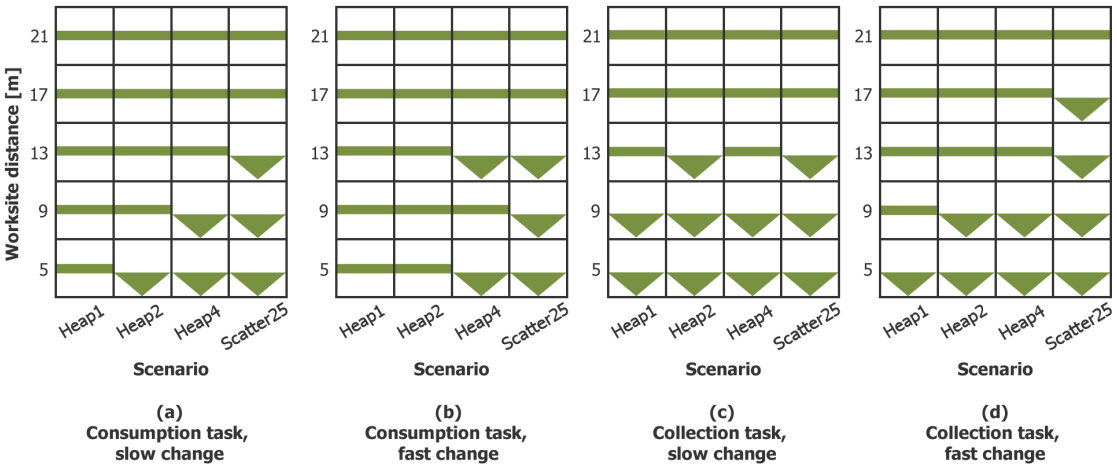


Figure 7.14: The effect of Opportunism on the performance of 25-robot bee swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix F, Figure F.28.

7.3 Summary and discussion

The Opportunism add-on control strategy causes a robot to adopt a worksite with a higher utility than that of a worksite that the robot is currently working on. Solitary robots are mostly unaffected by this strategy, since a robot that does not communicate with other members of the swarm rarely encounters more than one worksite during its work cycle.

On the other hand, in swarms that use communication, robots exchange information about the encountered worksites, meaning that they are able to compare multiple worksites fairly often. Opportunism positively affects local broadcasters in scenarios where worksites are numerous, since it allows the robots to exploit more profitable worksites, i.e., those that have larger volumes, instead of following information to worksites that other members of the swarm are already subscribed to. However, since Opportunism requires frequent communication between robots, it can lead to transmission of erroneous worksite locations to recruits in scenarios where congestion around worksites cannot be avoided, for example when worksites take a long time to deplete. This damages the swarm performance in the collection task when Heap worksites are very close to the base.

Bee swarm robots exchange worksite information in the base, which causes the preference for a worksite with the highest utility to be much more contagious, compared to when local broadcasters use Opportunism. Since the majority of bee swarm robots that use

Opportunism concentrate on a single worksite, congestion is significantly increased and the performance is lower than that of bee swarms without Opportunism.

The results presented here suggest that Opportunism can improve swarm performance as a result of exploitation of more profitable worksites, provided that the spread of information about worksite utility is sufficient but also regulated. In local broadcaster swarms, opportunistic behaviour is regulated by the limited communication range of robots, which prevents a large group of robots to receive and spread the same information. However, when robots communicate with each other in a small designated area, as it is the case in bee swarms, spreading of information is uncontrolled, which causes all robots to make the same decision about where to work, preventing both exploitation and exploration.

Behaviour similar to Opportunism in bee-inspired swarms, that needed to choose between worksites with different utilities in environments where worksite utilities changed periodically, was previously explored in (Pitonakova et al., 2016a, see Appendix I). In line with the results presented above, it was shown that bee swarms employing opportunistic behaviour (called "begger" behaviour in the paper), perform worse than other swarms due to their strong commitment to a small portion of worksites.

It is interesting that numerous studies of honey bee foraging, contrary to the results presented here, reported that exchanging information about nectar profitability among worker bees *improves* the ability of bee colonies to forage efficiently, because it is allows the colonies to react to rapid changes in nectar quality of different flower patches (De Marco and Farina, 2003; Farina et al., 2005). This discrepancy between the results from experiments with animals and with robots points to the importance of performing in-depth comparative studies of biologically inspired robot control algorithms. Animals often exhibit a particular behaviour, for example sharing of nectar profitability information, *in addition to* a large number of other self-regulatory and communication behaviours. For example, bees also regularly scout the environment in order to discover new flower patches (Seeley, 1994) and they periodically check the profitability of old abandoned patches (Granovskiy et al., 2012). The colony also monitors and maintains a healthy nectar intake in order to prevent energy wastage resulting from congestion in the nest (Anderson and Ratnieks, 1999; Gregson et al., 2003). However, as the results presented above indicate, opportunistic behaviour can lead to deterioration in swarm performance if it is used on robots without additional self-regulatory mechanisms.

Chapter 8

The Anticipation add-on strategy

Overcommitment to worksites, and the resulting congestion near them, can prevent robots from working and from scouting the environment, as well as from discovering when a worksite that they are subscribed to becomes depleted. This difficulty manifests itself in the form of high misplacement and opportunity costs paid by a swarm. Overcommitment damages swarm performance especially when there are a lot of undiscovered worksites in the environment and when the environment is dynamic, i.e., when worksites are only available for a limited amount of time.

It is thus important for swarms in such environments to be able to spread themselves across multiple worksites proportionate to the amount of resource available in the worksites. In this chapter, the *Anticipation* add-on control strategy, which is designed to prevent overcommitment to worksites with low volumes, is explored. Robots equipped with this add-on strategy can anticipate that their worksite will produce a low return or no return in the near future and abandon the worksite in order to search for a new one. As a result, congestion around worksites that are nearly depleted clears gradually, allowing robots to search the environment for more profitable sites. It is expected that:

Hypothesis 8.1. Anticipation decreases the amount of misplacement and opportunity cost paid by a swarm and thus increases the swarm's performance.

In order to allow robots to anticipate that a worksite is likely to have a low return, they evaluate how the worksite utility, U_W (see Chapter 7, Equations 7.1 and 7.2), changed since the last time it was measured. The change in U_W is expressed as the *relative worksite utility*, η_W:

$$\eta_W = \min\left(1.0, \frac{U_W}{U'_W}\eta'_W\right) \tag{8.1}$$

where U'_W and η'_W are the worksite utility and relative worksite utility measured previously. The value of η_W decreases while a robot is depleting a worksite, since the volume

of the worksite decreases and $U'_W > U_W$. When a robot discovers a new worksite during scouting, its η_W is set to 1.0. When a robot is recruited, it adopts the η_W measured and transmitted by the recruiter.

All three basic control strategies, solitary, local broadcaster and bee, may be equipped with the Anticipation add-on. The add-on strategy affects the behaviour of robots as follows:

- A robot evaluates the U_W and η_W of the worksite that it currently performs work at. In the consumption task, U_W and η_W are measured each second while a robot is working. In the collection task, U_W and η_W are measured before a robot leaves the worksite to unload resource in the base.
- After a measurement is made, the robot decides to abandon its worksite with an abandonment probability $p(A) = 1 - \eta_W$. The abandonment probability thus increases while a worksite is being depleted. Robots from the solitary and local broadcaster swarms become scouts upon abandoning a worksite. Bee swarm robots return to the base and become observers instead.

In the following sections, the impact of Anticipation on the characteristics and the performance of swarms is evaluated. It is shown that Anticipation has a positive effect on swarm performance when robots that abandon their worksites as a result of low η_W can find information about new worksites quickly enough. However, it is also important to prevent all uninformed robots from concentrating on the same worksite at the same time. Hypothesis 8.1 is thus only supported when bee swarms with Anticipation are used in the dynamic collection task. In other cases, Anticipation decreases the performance of swarms, as worksite abandonment leads to information loss, and consequently to a higher amount of uncertainty cost incurred by robots.

8.1 Swarm characteristics

8.1.1 Scouting efficiency

Even though robots often abandoned worksites before they were depleted when Anticipation was turned on, the time of the first worksite discovery (see Section 6.1.1) was usually not significantly affected, since the number of scouts was not sufficiently higher when the environment changed.

8.1.2 Information gain rate

As a result of Anticipation, there were more scouts available in the environment when solitary and local broadcaster swarms were used. Consequently, even though the time of first worksite discovery was not affected (see Section 8.1.1), the information gain rate, i, (see Section 4.2.3) of these swarms was higher when Anticipation was used, compared to when it was not used. The positive effect of the add-on strategy on i for solitary and local broadcaster swarms was significant in the collection task when the work arena was most densely populated by robots and worksites, i.e., when medium-sized and large swarms ($N_R \geq 25$) were used in Scatter25 scenarios with a small D (see Figures 8.1 and 8.2 for results from 25-robot swarms and Appendix G, Figures G.1 – G.4 for results from 10- and 50- robot swarms, respectively). Solitary swarms were more positively affected, since they did not use recruitment, meaning that the robots were more spread out in the environment when it changed.

The information gain rate of bee swarms was not affected by Anticipation, since bee robots that abandoned their worksites returned to the base and did not scout immediately. There was also no clear trend in the effect of Anticipation on i of any of the investigated control strategies in the consumption task.

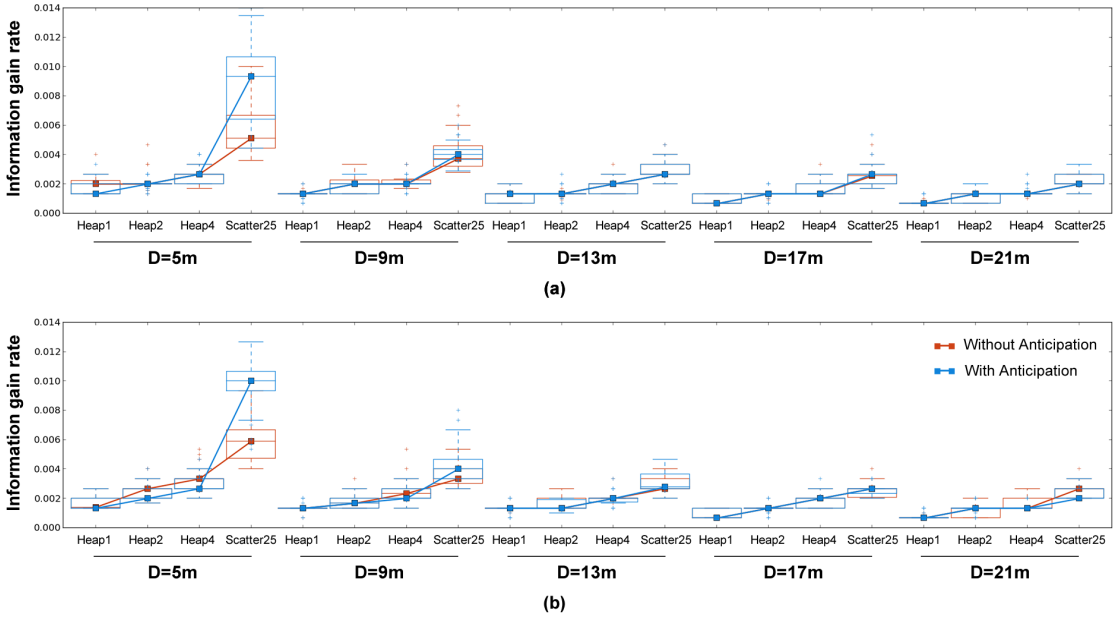


Figure 8.1: Information gain rate, i, of 25-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The i of solitary swarms was higher with Anticipation in Scatter25 scenarios with a small D.

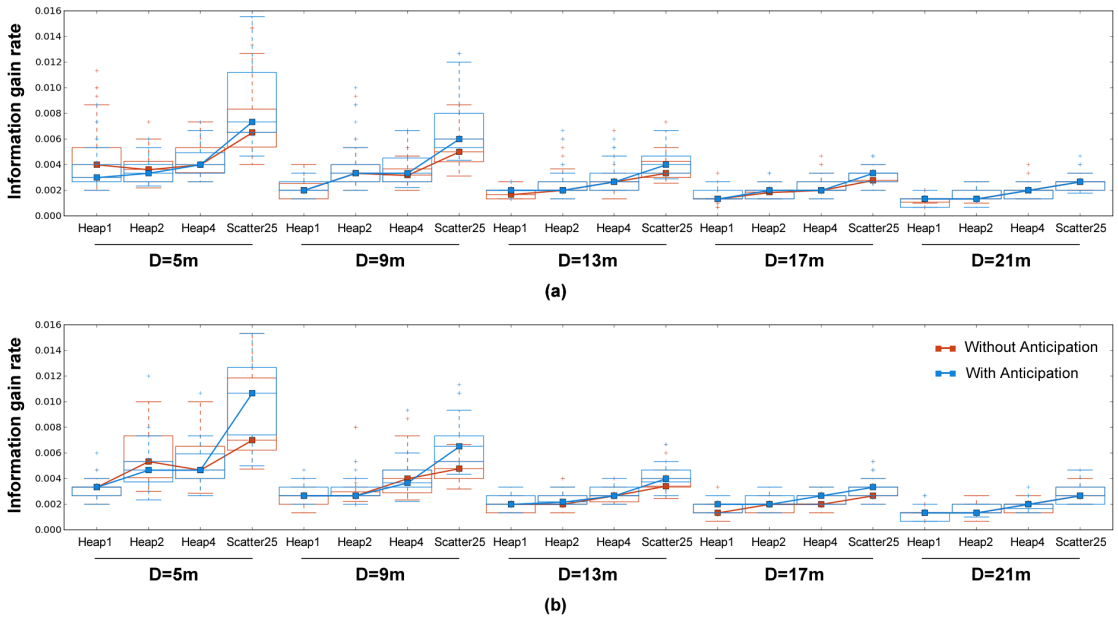


Figure 8.2: Information gain rate, i, of 25-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The i of local broadcasters was higher with Anticipation in Scatter25 scenarios when $D \leq 13$m and the environment changed quickly.

8.1.3 Tendency to incur misplacement cost

The misplacement cost coefficient, m, (see Section 4.3.2) of solitary swarms was equal to 0 in the dynamic consumption task, regardless of whether Anticipation was used. Similarly, the median m of all swarms in the dynamic collection task remained equal to 1 when Anticipation was used.

In the dynamic consumption task, the misplacement cost coefficient of local broadcasters was higher with Anticipation in scenarios where worksites were being depleted slowly and where worksite abandonment thus lead to a higher number of scouts and a higher probability of robots being recruited by other working members of the swarm. This was the case in Heap environments when 10 or 25 robots were used and when worksites were further away from the base, while the environment changed slowly (see Figure 8.3 for results from 25-robot swarms and Appendix G, Figures G.5 and G.6 for results from 10- and 50- robot swarms, respectively). On the other hand, the m of local broadcasters was smaller when Anticipation was used in environments where congestion normally prevented many robots from accessing worksites that they were recruited to. Anticipation decreased the m of local broadcasters when the swarms were large ($N_R = 50$) and when Heap worksites were close to the base while the environment changed slowly (see Appendix G, Figure G.6). In these cases, probabilistic abandonment of worksites decreased the amount of congestion near them, meaning that new recruits could reach the worksites more easily and thus incurred a lower misplacement cost.

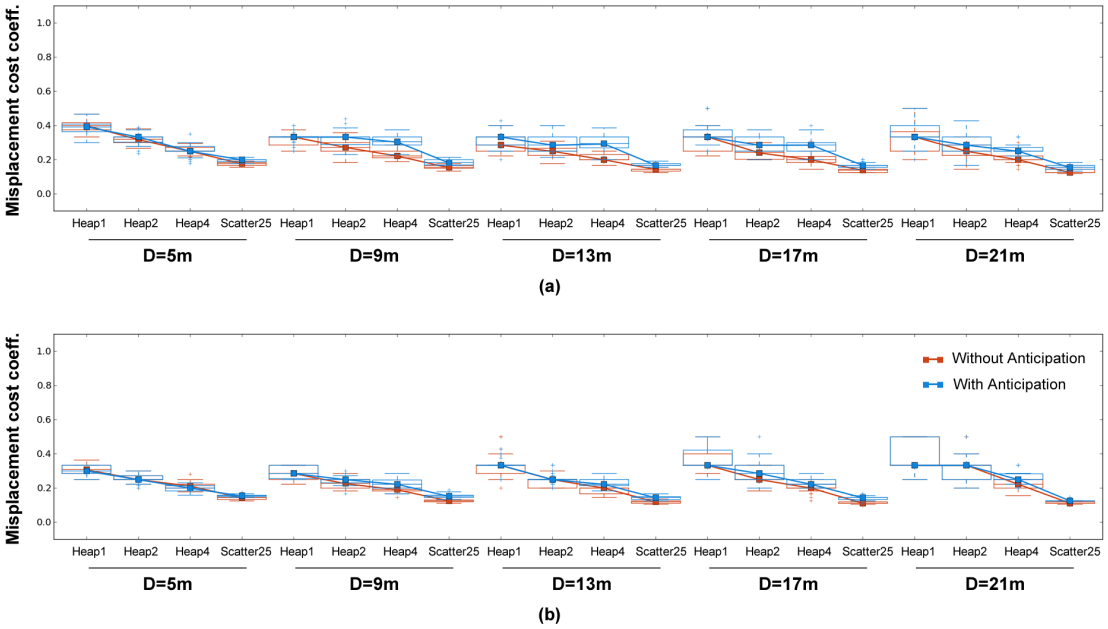


Figure 8.3: Misplacement cost coefficient, m, of 25-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The m of local broadcasters was higher with Anticipation in Heap2, Heap4 and Scatter25 scenarios when $D \geq 9$m and when the environment changed slowly.

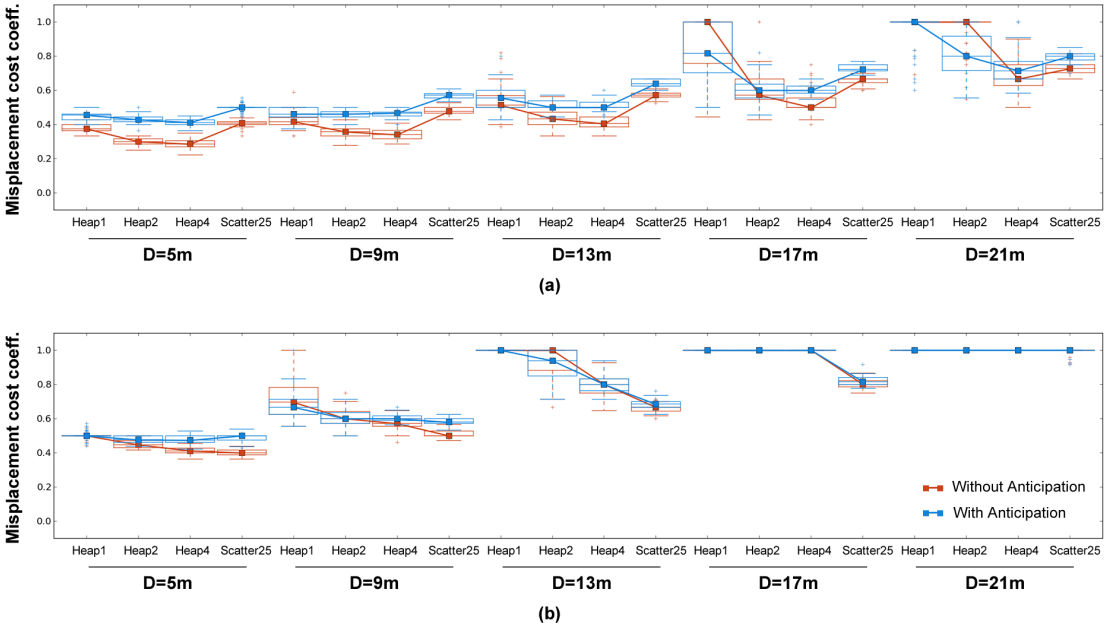


Figure 8.4: Misplacement cost coefficient, m, of 25-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was higher with Anticipation in most scenarios, especially when worksites were close to the base and when the environment changed slowly.

The misplacement cost coefficient of bee swarms was more strongly affected by Anticipation than that of local broadcasters, especially when worksites were close to the base and when the environment changed slowly. The m of bee swarms was higher with, compared to without, Anticipation in the majority of the explored environments (see Figure 8.4 for results from 25-robot swarms and Appendix G, Figures G.7 and G.8 for results from 10- and 50- robot swarms, respectively). Since abandonment of worksites caused more bee swarm robots to become observers, there were more robots available in the base to recruit. The recruited robots incurred misplacement cost, C_M, while traveling to worksites advertised by informed members of the swarm.

8.1.4 Tendency to incur opportunity cost

Solitary robots incurred a smaller amount of opportunity cost, C_O, (see Section 4.3.3) when they used Anticipation in the dynamic collection task (see Figure 8.5 for results from 25-robot swarms and Appendix G, Figures G.9 and G.10 for results from 10- and 50- robot swarms, respectively), since some robots abandoned worksites early and thus did not incur C_O after the environment changed. This was especially apparent when the environment changed slowly, i.e. when more robots were normally subscribed to worksites at the end of a change interval when Anticipation was not used. The C_O of solitary swarms remained equal to 0 in the dynamic consumption task.

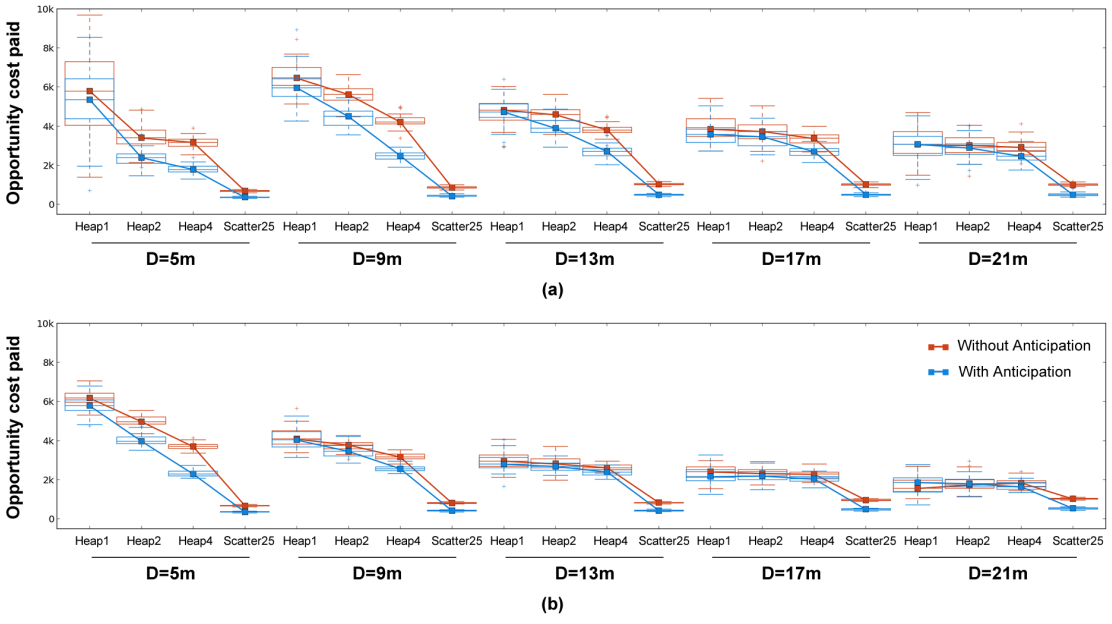


Figure 8.5: Opportunity cost, C_O, of 25-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by solitary swarms was smaller with Anticipation in most of the scenarios, especially when the environment changed slowly.

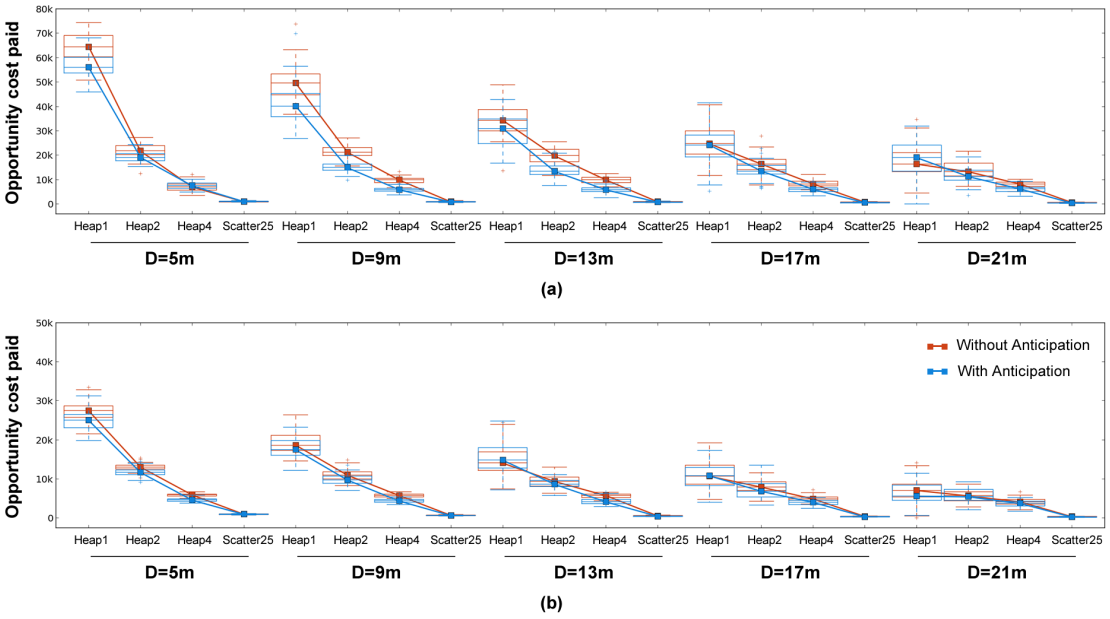


Figure 8.6: Opportunity cost, C_O, of 50-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by local broadcasters was smaller with Anticipation in most Heap scenarios when $D \leq 13$m, especially when the environment changed slowly.

The tendency of local broadcasters to incur C_O decreased when Anticipation was turned on in the dynamic consumption task, in scenarios where congestion normally prevented robots from accessing worksites. This was most notably the case when large swarms ($N_R = 50$) were used in slowly changing environments with worksites that were close to the base (see Figure 8.6 for results from 50-robot swarms and Appendix G, Figures G.11 and G.12 for results from 10- and 25- robot swarms, respectively). On the other hand, the amount of C_O paid by local broadcasters remained similar with, compared to without Anticipation in the dynamic collection task, where congestion was not as severe since robots periodically left worksites to unload resource in the base.

The amount of C_O paid by bee swarms was smaller or similar when Anticipation was used in the dynamic collection task, but higher in the dynamic consumption task (see Figures 8.7 and 8.8 for results from 25-robot swarms and Appendix G, Figures G.13 – G.16 for results from 10- and 50- robot swarms, respectively). In both tasks, worksite abandonment caused more observers to be present in the base, and thus led to more recruitment (see Section 8.1.3). However, in the consumption task, all the observers were usually recruited to a single worksite, while in the collection task, different observers were recruited to different worksites. The reason for this difference can be explained by considering how many recruiters with information about different worksites were present in the base at a given point in time. During consumption, recruitment only happened when a scout made a new discovery, meaning that there was usually a very small number

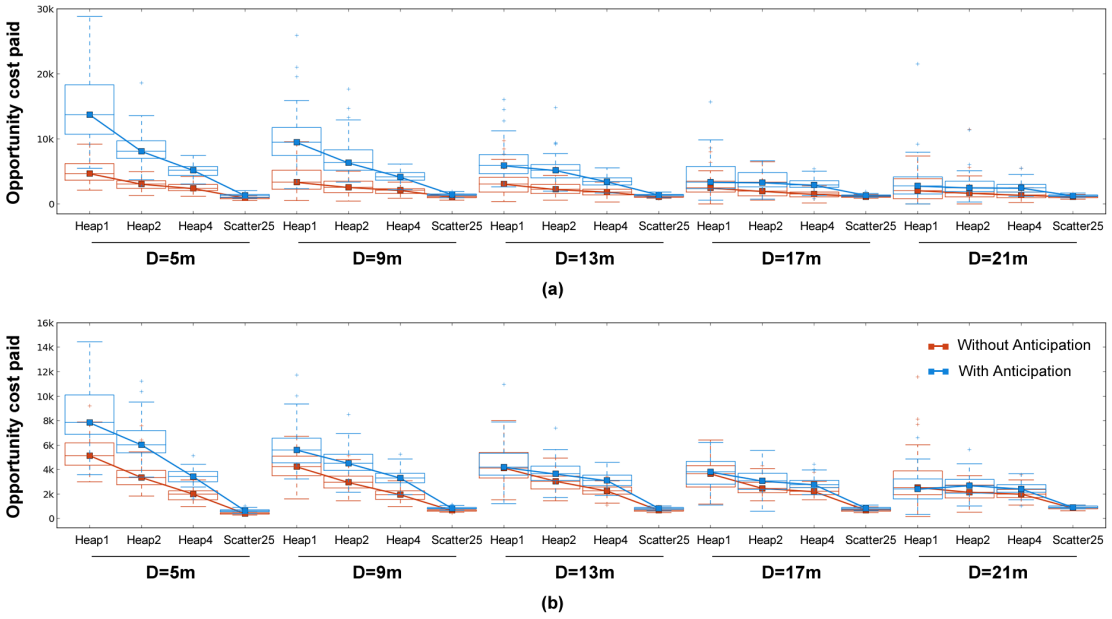


Figure 8.7: Opportunity cost, C_O, of 25-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by bee swarms was larger with Anticipation in most scenarios, especially when D was small or when the environment changed slowly.

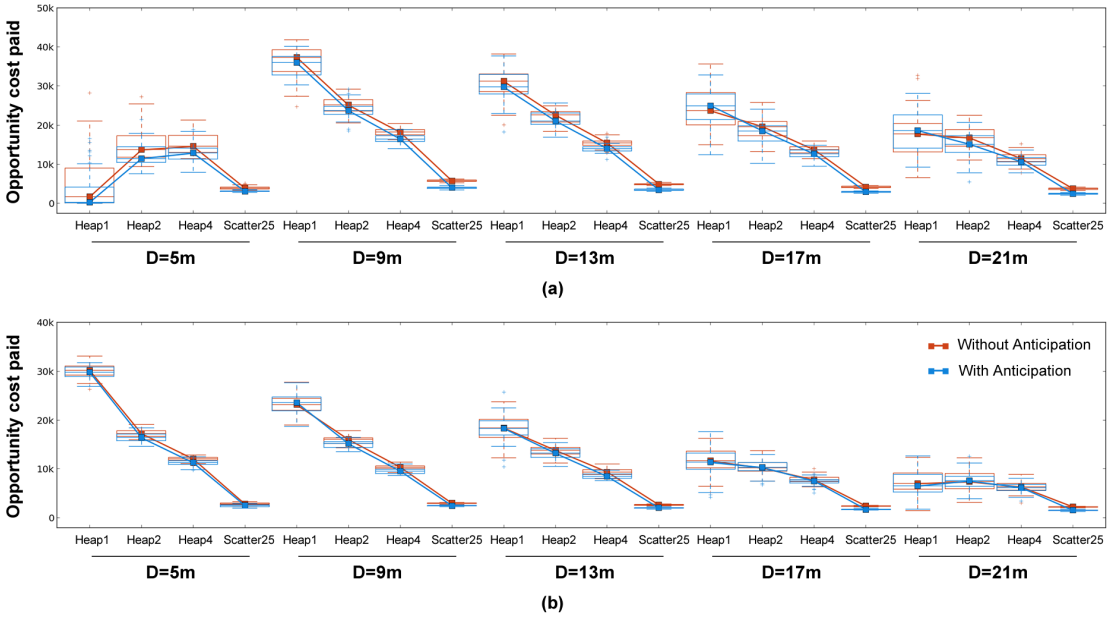


Figure 8.8: Opportunity cost, C_O, of 25-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by bee swarms was smaller with Anticipation in some Heap4 and Scatter25 scenarios. The C_O paid was unaffected by Anticipation in other scenarios.

of recruiters in the base. This caused observers, if they were recruited, to be sent to a small number of new locations. Furthermore, if a recruiter happened to have information about a worksite that disappeared while the robot was in the base, many observers were sent to a wrong location and thus incurred a large amount of C_O. On the other hand, in the collection task, every worker periodically returned to the base to unload resource, meaning that there was information about a larger number of worksites available in the base. Consequently, recruited observers were divided up more evenly across different working locations during this task and the swarm incurred a smaller amount of C_O.

8.2 Swarm performance

Anticipation had the following impact on the basic control strategies:

- In solitary and local broadcaster swarms, the number of scouts in the environment increased as a result of abandonment of nearly depleted worksites, leading to a higher information gain rate in the dynamic collection task, Scatter25 environments. Information gain rate did not change significantly in bee swarms, since bee swarm robots that abandoned worksites returned to the base and became observers.
- Since there was a smaller number of robots working on worksites that were about to be depleted or disappear from the environment at the end of a change interval, solitary swarms paid a smaller amount of opportunity cost, C_O, when they used Anticipation in the dynamic collection task. Their $C_O = 0$ in the dynamic consumption task.
- Similarly, as a result of lower commitment to worksites and decreased congestion near them, local broadcasters paid a smaller amount of C_O and C_M in the dynamic consumption task when the swarms were large.
- Anticipation led to more recruitment in bee swarms, since there were more observers in the base compared to when Anticipation was not used. In the consumption task, recruiters arrived at the base less often, meaning that if recruitment took place, many observers usually followed to a single worksite. This increased the amount of C_O paid by bee swarms that used Anticipation. On the other hand, in the collection task, recruiters from different worksites were usually present in the base at the same time, causing observers to be recruited to different worksites. The amount of C_O paid decreased when Anticipation was turned on in the dynamic collection task as a result of early worksite abandonment.

Figures 8.9 – 8.11 show the effect of Anticipation on the performance of 25-robot swarms. Results for 10- and 50-robot swarms are shown in Appendix G, Figures G.17 – G.22. As

was expected by Hypothesis 8.1, the performance of bee swarms that use Anticipation improves in the dynamic collection task in scenarios where new information is easy to discover, i.e., when the number of worksites is large or when worksites are close to the

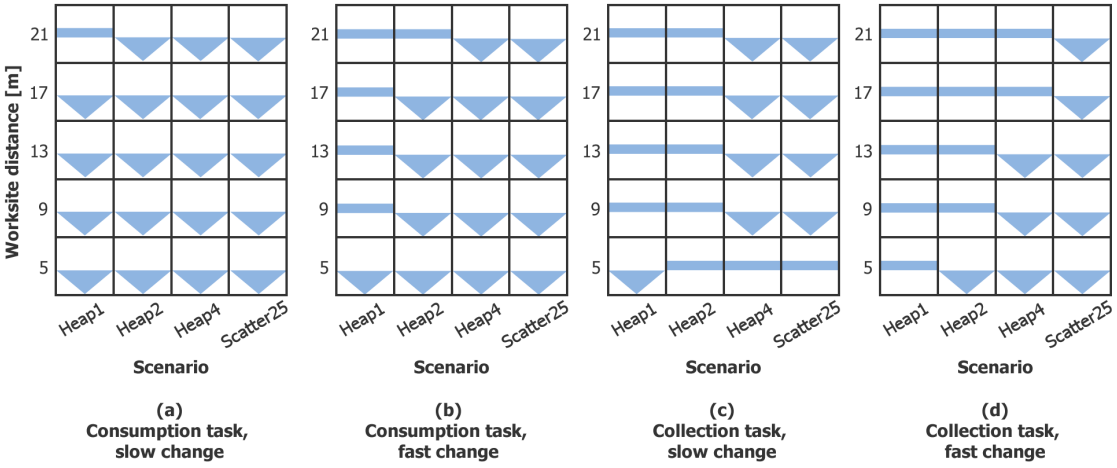


Figure 8.9: The effect of Anticipation on the performance of 25-robot solitary swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix G, Figure G.24.

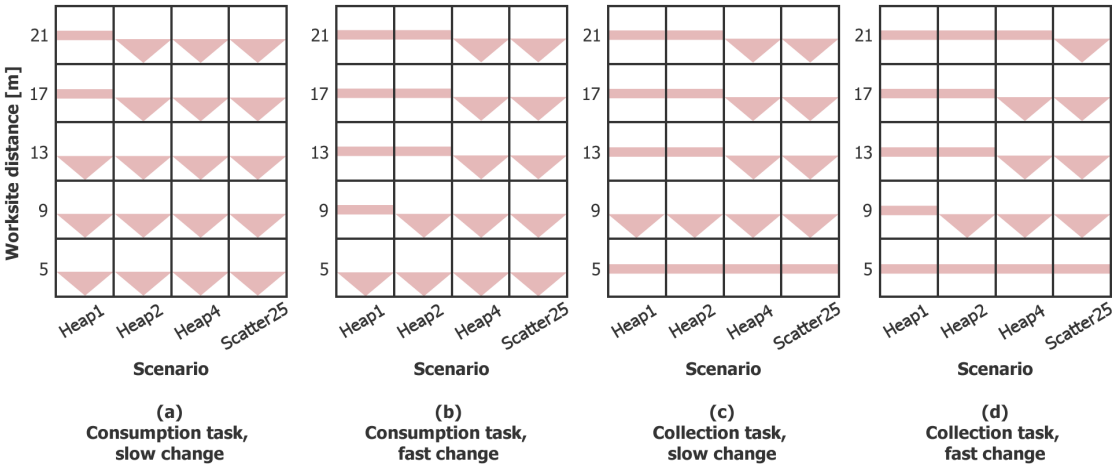


Figure 8.10: The effect of Anticipation on the performance of 25-robot local broadcaster swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix G, Figure G.27.

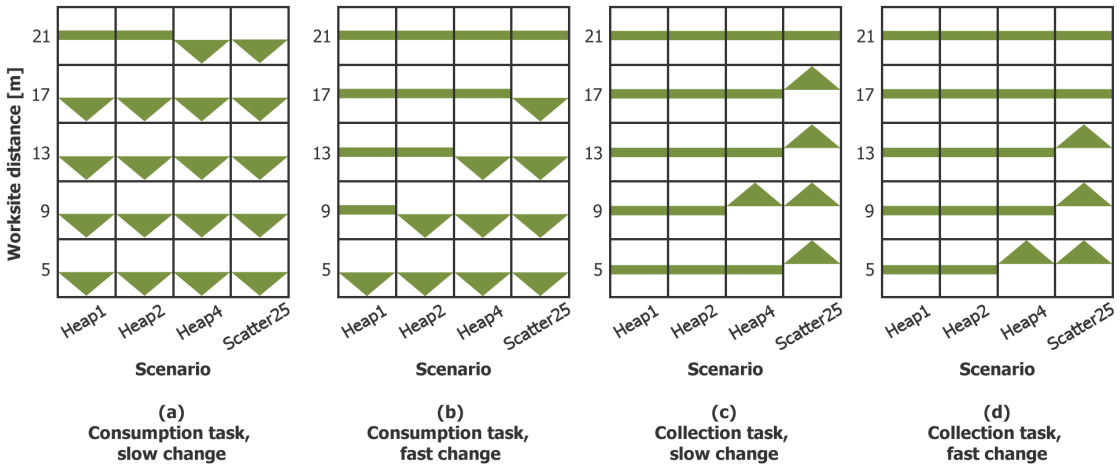


Figure 8.11: The effect of Anticipation on the performance of 25-robot bee swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Appendix G, Figure G.30.

base (Figure 8.11c,d). In these cases, robots collecting resource gradually abandon a worksite and are then recruited to different worksites with higher volumes. Anticipation has a more positive effect when the environment changes slowly (see Appendix G, Figures G.29 – G.31), as abandonment and recruitment can take place more often in each change interval.

On the other hand, the performance of swarms is often negatively affected by the add-on behaviour. Solitary and local broadcaster swarms collect significantly less resource when they use Anticipation in the majority of the explored scenarios, despite of being able to discover more worksites and incurring a smaller amount of opportunity, and in the case of local broadcasters, misplacement, costs. Similarly, bee swarms perform significantly worse when they use Anticipation in the dynamic consumption task.

The unexpected decrease in swarm performance is caused by information loss. When robots unsubscribe from worksites, the swarms possess less information about the environment. Significant information loss is apparent from comparing the amount of uncertainty cost, C_U (see Section 4.3.2), paid by the swarms with and without Anticipation (see Figures 8.12 – 8.17 for results from 25-robot swarms and Appendix G, Figures G.32 – G.43 for results from 10- and 50- robot swarms, respectively). Solitary and local broadcaster swarms pay a higher amount of C_U in most scenarios. More C_U is paid in the consumption, compared to the collection task, since the relative worksite utility is evaluated more often during the consumption task and worksites are thus abandoned more frequently. Furthermore, Anticipation increases the amount of C_U paid by a larger amount when swarms are smaller, i.e., when new worksite discoveries are less probable.

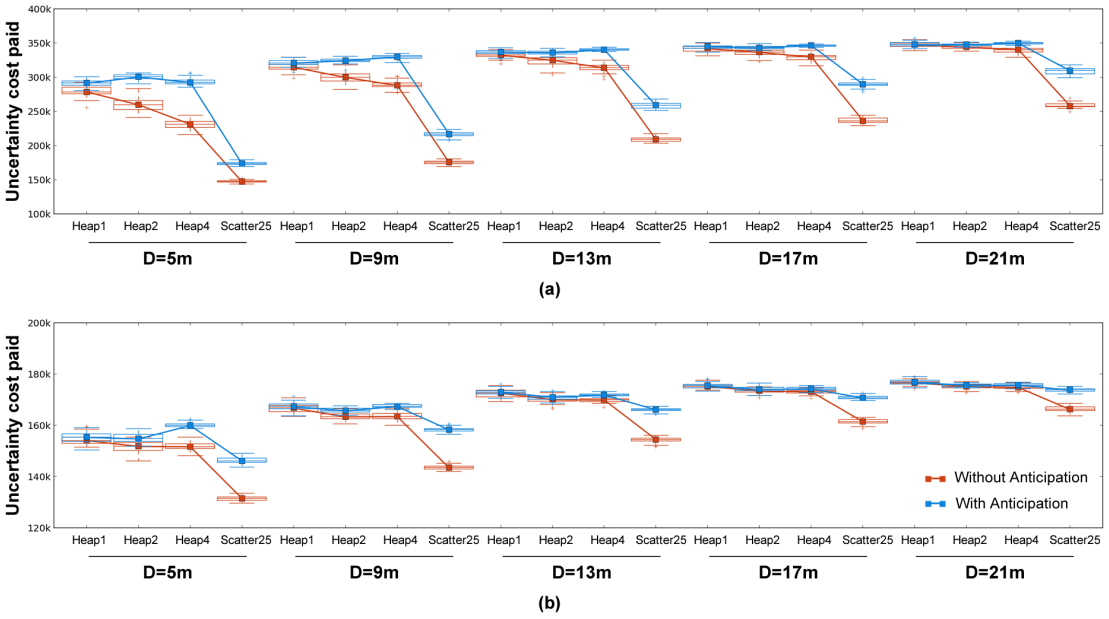


Figure 8.12: Uncertainty cost, C_U, of 25-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by solitary swarms was larger with Anticipation in most scenarios.

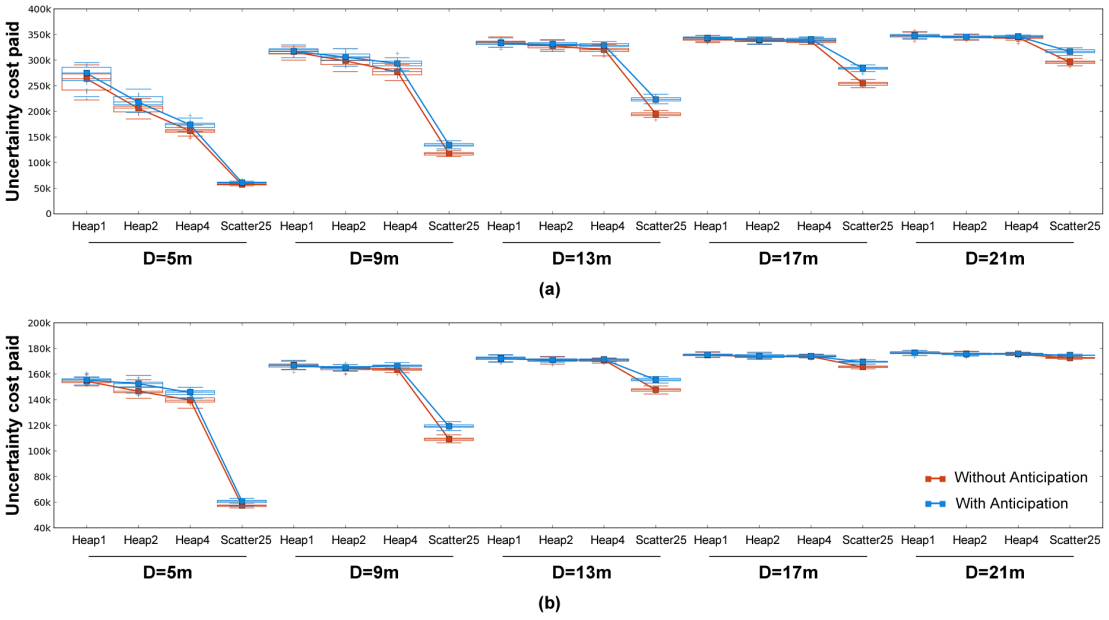


Figure 8.13: Uncertainty cost, C_U, of 25-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by solitary swarms was higher with Anticipation in Scatter25 scenarios and some Heap scenarios.

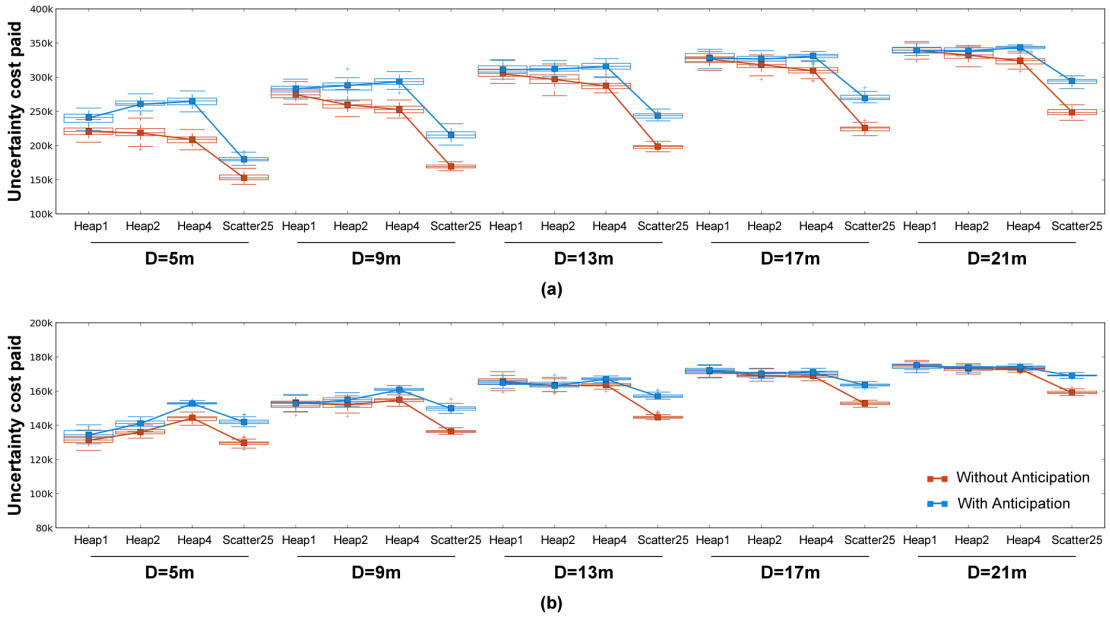


Figure 8.14: Uncertainty cost, C_U, of 25-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by local broadcasters was larger with Anticipation in most scenarios.

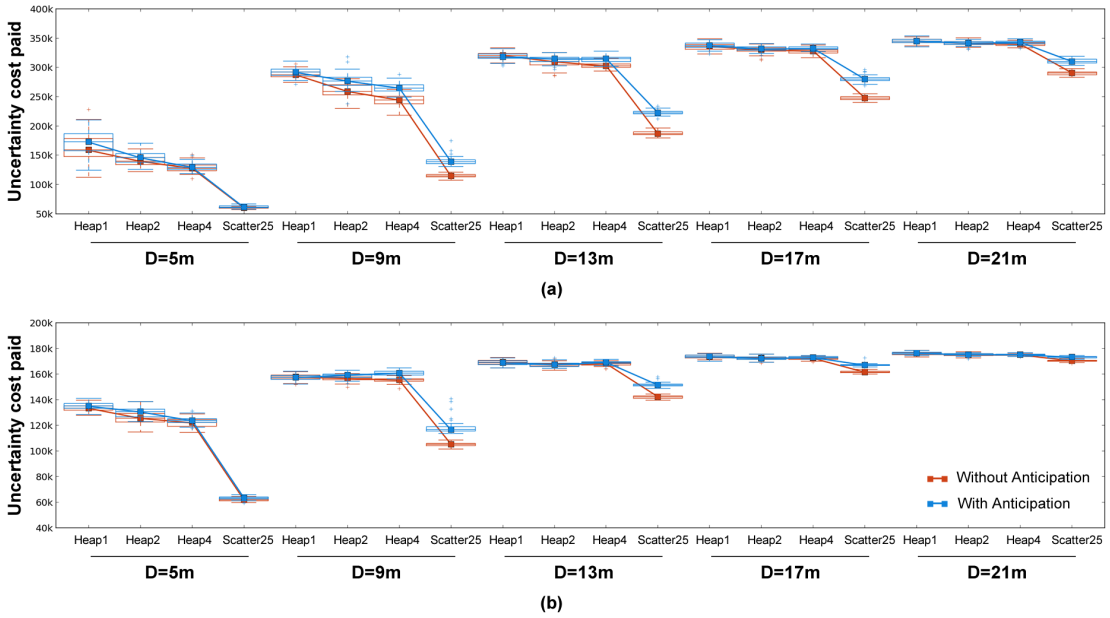


Figure 8.15: Uncertainty cost, C_U, of 25-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by local broadcasters was larger with Anticipation in Scatter25 scenarios and some Heap4 scenarios.

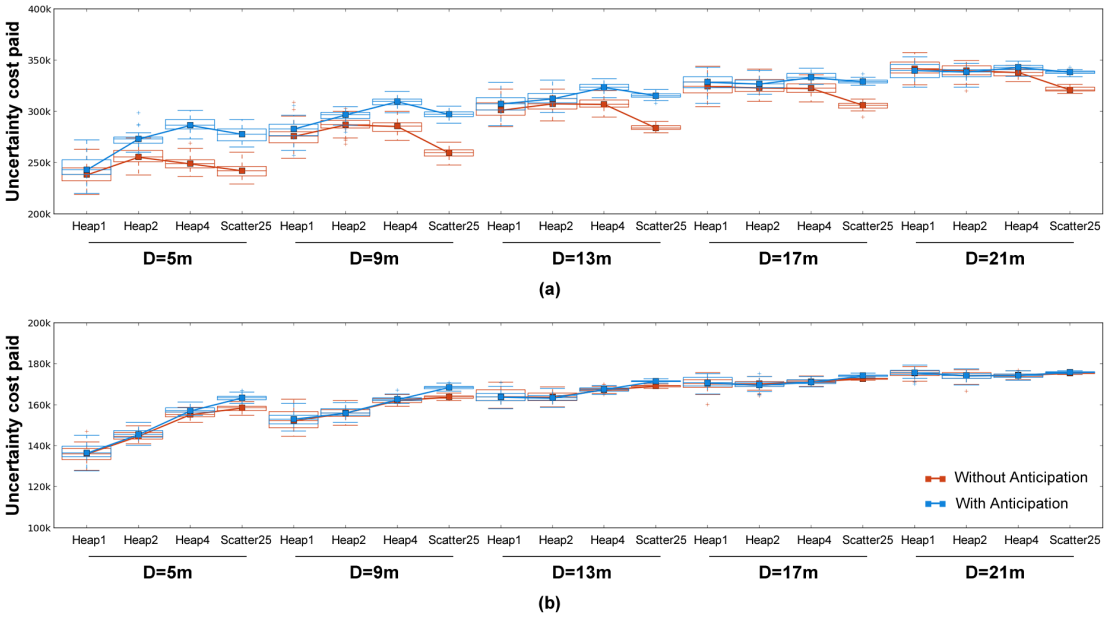


Figure 8.16: Uncertainty cost, C_U, of 25-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by bee swarms was larger with Anticipation in most scenarios when the environment changed slowly. When the environment changed quickly, the amount of C_U paid was larger in most Scatter25 scenarios.

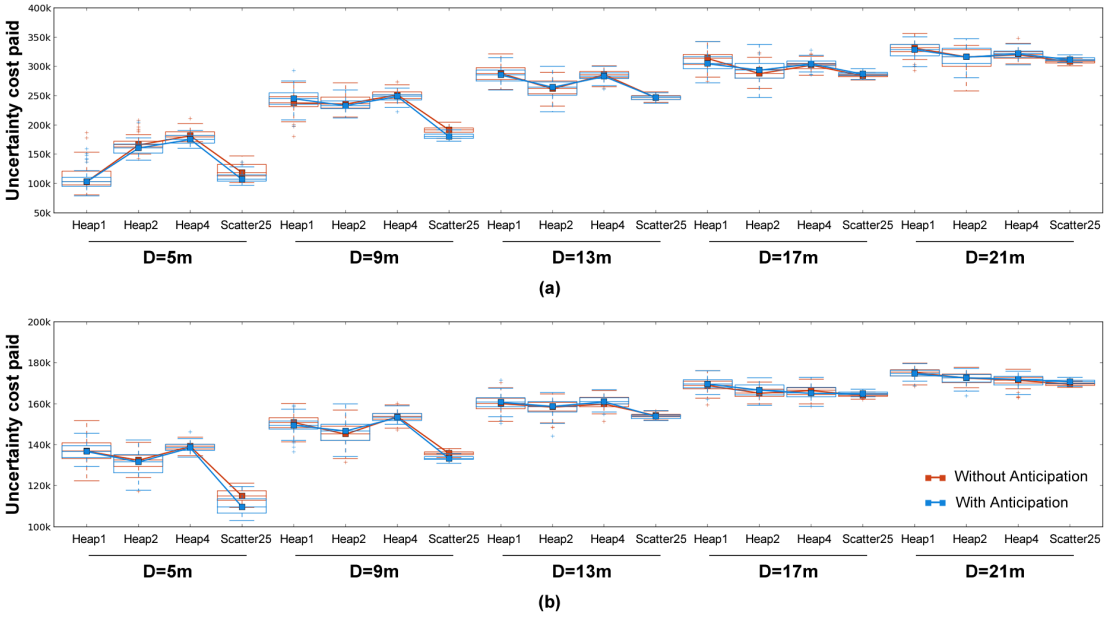


Figure 8.17: Uncertainty cost, C_U, of 25-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by bee swarms is smaller with Anticipation in some Scatter25 scenarios, while it remained unaffected in other scenarios.

On the other hand, bee swarms that use Anticipation pay a similar or a smaller amount of C_U, compared to bee swarms that do not use the add-on strategy, in some scenarios during the dynamic collection task (see Figure 8.17 and Appendix G, Figures G.42 and G.43), which allows them to improve their work performance.

8.3 Summary and discussion

In dynamic environments, where worksites are available for a limited amount of time, it is important for a swarm to continuously monitor the work arena and to spread itself among worksites based on how much resource there is available in them. The Anticipation add-on strategy allows a swarm to do that by causing robots to abandon worksites that are likely to be depleted soon. As a result, more scouts roam the work arena and a swarm is more likely to discover new sources of reward.

However, since worksite abandonment leads to information loss, it is important for robots to be able to find new information quickly. Robots from solitary and local broadcaster swarms are unable to do so, as their probability to discover worksites is subject to random movement of robots and random structure of the environment. These swarms thus often suffer from performance degradation when they use Anticipation.

Apart from the ability of robots that abandon worksites to discover a new reward source quickly, it is also important to prevent physical and exploitational interference between robots. Anticipation thus only increases the performance of bee swarms that perform the dynamic collection task in easy environments. In these cases, robots that unsubscribe from nearly depleted worksites can travel to the base and are likely to get recruited by other, informed, members of the swarm to various parts of the work arena and to continue work where more reward is available. On the other hand, in the consumption task, during which bee swarm robots do not return to the base to unload resource, only a small number of recruiters is usually available in the base. As it is indicated by the high amount of opportunity costs that bee swarms with Anticipation pay, robots are mostly recruited to a very small number of worksites, which causes congestion and consequently a worse performance, compared to when Anticipation is not used.

Robot behaviour very similar to the Anticipation add-on behaviour investigated here was previously described in (Pitonakova et al., 2016a, see Appendix I). It was shown that Anticipation improved the performance of bee-inspired swarms in dynamic environments with variable worksite utilities. Swarms equipped with anticipation could discover more quickly that an environment had changed than swarms with other control strategies. In the setting explored in the paper, bee-inspired swarms were performing the collection task, and thus periodically met in the base. Therefore, like in the bee swarms explored here, there was a high probability that a robot would find useful information in the base after it abandoned a worksite as a result of its low relative utility.

Honey bees also evaluate the change in nectar profitability of a flower patch that they forage from, which allows them to maintain positive energy income while the weather changes (Seeley et al., 1991; Biesmeijer and De Vries, 2001). When a bee forager returns to the base, it performs a shorter recruitment dance, or no recruitment, when its patch quality is poor. Such a bee is also more likely to abandon its own patch completely and to be recruited by another forager to a patch that is more profitable. In a similar fashion to the bee-inspired robots investigated here, honey bees can afford to abandon poor flower patches because information about where to forage from is often available in their nest.

Chapter 9

Design patterns

It was demonstrated in Chapters 4– 8 that the performance of a swarm depends on a number of environmental and swarm characteristics. Consequently, the relative success of one robot control strategy over another control strategy is a function of these characteristics. Robot swarm designers should thus be able to make decisions about what robot control algorithms to implement based on the available mission facts. To this end, design patterns can provide a useful set of guidelines.

A single design pattern can be understood as a "template" for a particular part of a robot control algorithm. For example, a design pattern might suggest how information about worksites is exchanged between robots. Multiple design patterns can be combined in order to create a robot control algorithm that is suitable for a given mission and that can be implemented on all or on a subgroup of robots in a swarm. Because they include not only description of robot behaviour, but also discuss consequences of that behaviour on macro-level swarm characteristics and performance, design patterns aid decision-making of developers by providing solutions that work well in particular missions. In this thesis, only design patterns for homogeneous robot swarms are considered. Therefore, any description of their consequences on swarm-level behaviour is written with the assumption that all robots execute the same control algorithm. However, in general, design patterns could also be applied in heterogeneous swarms, where only a sub-group of robots would behave according to a particular design pattern. See Section 10.2 for a further discussion on heterogeneous robot swarms.

The following important mission characteristics, that need to be considered when selecting appropriate design patterns, have been identified throughout this thesis:

- **Worksite density**, i.e., how probable it is that a worksite can be found by a robot, given its current location
- **Worksite volume**, i.e., how quickly a worksite gets depleted when robots perform work on it

- **Misplacement of reward from worksites**, i.e., whether robots need to travel away from worksites in order to obtain reward, for example to drop off resource in the base during the collection task
- **Dynamics of the environment**, i.e., whether the worksite characteristics, such as their location, change over time. In dynamic environments, worksite characteristics are temporary and reward needs to be extracted from worksites as quickly as possible.

Furthermore, there are four characteristics of a swarm that result from its control strategy:

- **Scouting efficiency** (see Section 4.2.1), related to how quickly the swarm can discover new worksites
- **Information gain rate,** i, (see Section 4.2.3), related to how quickly information enters into the swarm and spreads within it as a result of scouting and communication between robots
- **Tendency to incur misplacement cost,** C_M, (see Section 4.3.2), related to how much time is spent by robots approaching worksites that they have become subscribed to, as well as the number of such robots
- **Tendency to incur opportunity cost,** C_O, (see Section 4.3.3), related to how long it takes robots to discover that the worksites that they are subscribed to have disappeared from the environment, as well as the number of such robots

The first two swarm characteristics describe how information is obtained by robots and how it is shared between them. The last two characteristics describe how efficiently a swarm can transform information that it has gained into the reward that it was created to earn. The Information-Cost-Reward (ICR) framework, introduced in Section 4.4.5 and depicted again in Figure 9.1, expresses the relationship between the swarm and the environmental characteristics, as well as between the swarm characteristics themselves. Under this framework, a swarm is understood as a single entity capable of decentralised cognition at the level of the collective that emerges from information processing, actions and interactions of individual robots.

When the environment is static and worksites are difficult to find, control strategies with a high information gain rate are suitable. However, since high information gain rate is usually achieved by communication between robots, it can be associated with an increase in the tendency of swarms to incur misplacement and opportunity costs. Consequently, when worksites are relatively easy to discover, swarms with a high i tend to have lower performance than swarms with a lower i (see Chapters 4 and 5). Furthermore, when the environment is dynamic, i.e., when new information is generated in the environment over

time, a good scouting efficiency and a low tendency to incur costs are more important (see Chapter 6). Finally, costs can be ameliorated by the environment. For example, the negative effect of misplacement cost during the collection task is smaller in swarms where robots exchange information at the location where resource needs to be dropped off (see Chapter 5).

In this chapter, general lessons learned from simulation experiments are formalised as design patterns. Each pattern captures a specific aspect of robot behaviour and provides a description of environmental conditions for which it is a suitable design choice. The methodology for design pattern creation is first introduced, followed by a catalogue of seven design patterns. Next, the rules for combining design patterns into control strategies are presented and examples are given. The chapter ends with a discussion of the relevant aspects of existing swarm robotic literature. It is shown that the design patterns introduced here can be found in robot control algorithms used in a broad range of robotic experiments and that the basic characteristics of these design patterns remain the same across different implementations. Finally, it is demonstrated that the methodology for design pattern creation established here can be applied in order to extend the design patterns catalogue and that the new design patterns can easily be combined with the existing ones. In Chapter 10, design patterns are considered in relation to other swarm

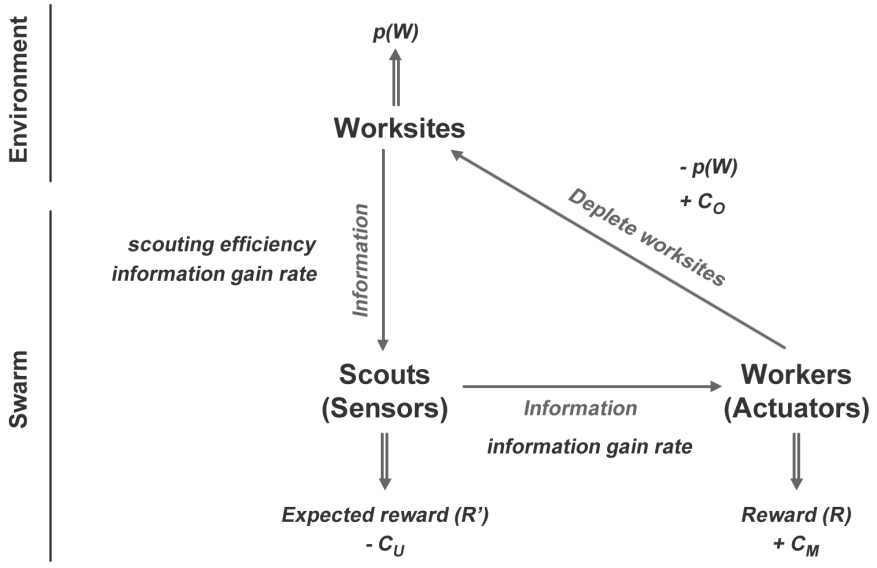


Figure 9.1: The swarm work cycle within the Information-Cost-Reward framework. Worksite distribution in the environment is characterised by the probability, $p(W)$, of a worksite being located at at a given point in space. Scouts search for and spread information about worksites. This process is affected by the swarm's scouting and communication strategies and produces expected reward, R', while reducing the swarm's uncertainty cost, C_U. Workers turn information into reward, R, after paying certain misplacement cost, C_M. They also alter the environment by depleting worksites, decreasing $p(W)$ and potentially increasing opportunity cost, C_O, paid by the swarm.

design methodologies, such as probabilistic finite state machine models and evolutionary algorithms, and relevant future research directions are identified.

9.1 Methodology

9.1.1 Design pattern creation

A total of nine robot control strategies were explored throughout Chapters 4 – 8. First, the basic control strategies, solitary, local broadcaster and bee, were experimented with separately. Two add-on strategies, Opportunism and Anticipation, were then explored in combination with the three basic strategies in order to find out whether they improved swarm performance in dynamic environments. In order to translate these strategies into design patterns, logic modules, each describing a unique aspect of a robot control algorithm, need to be identified. Inspired by the object-oriented design pattern principles (Gamma et al., 1994, p. 11–42), it is proposed here that a swarm robotic design pattern should:

- Describe a particular **stand-alone module of a robot control algorithm**, in terms of *robot behaviours*, relevant internal and external *data structures* and *relationships* between them. Such a module should satisfy a particular functional requirement and its description should be independent of other modules that deal with other requirements.
- Provide a description of **suitable environments and swarm tasks**, where the pattern is understood to be a suitable design choice
- Be possible to **combine** with other design patterns
- Be **implementation non-specific**, i.e., only describe high-level behaviour, rather than a particular algorithm or implementation[1]

The last point is especially important. In order to generate general knowledge from specific experiments, we need to dispose of implementation details, such as specific control algorithms or parameter values, and instead identify general patterns of robot behaviour and their implications on swarm performance. The Information-Cost-Reward framework is useful in this endeavour, as it describes robot behaviour in terms of "information flow" instead of "control code" and the result of the behaviour in terms of individual "costs" rather than simply "performance improvement" or "performance degradation". Descriptions of swarm behaviour that use the ICR framework are thus not dependent on

[1] This requirement is similar to the requirement of object-oriented design patterns for programming to an *interface* rather than an *implementation*, which leads to reusability and minimal implementation dependencies (Gamma et al., 1994, p. 30)

particular robot hardware or software and the results of the behaviour can be understood in a detailed manner and in relation to environmental characteristics.

An important step in creating a design pattern is to identify what *category* it should belong to. Categorising the set of processes that need to be formalised as a design pattern can assist in choosing what robot behaviours and data structure are relevant. Based on the performed experiments with the consumption and collection tasks, it is proposed here that a swarm robotic design pattern should belong to one of three categories, each answering a particular question:

- **Transmitter patterns:** What entity transmits information?
- **Exchange patterns:** Where and when is information shared?
- **Update patterns**: How is information updated?

Each design pattern description should include the following (similarly as in, e.g., Gamma et al., 1994; De Wolf and Holvoet, 2007; Gardelli et al., 2007; Fernandez-Marquez et al., 2013):

- Design pattern **name** and **category**
- A list of **suitable applications**
- Description of the robot **behaviours**
- **Dependencies** on other behaviours of the robot, including recommendations about which other design patterns it works effectively with
- A list of **parameters** associated with the robot behaviours, as well as, if possible, their impact on swarm performance
- A list of **consequences** that the design pattern has on swarm characteristics, expressed in terminology of the ICR framework. Since it is often not immediately obvious by looking at robot behaviour description, the design pattern consequences should identify how a particular micro-level routines affect the emergence of macro-level outcomes.

The design pattern name, category, suitable applications, behaviour description and consequences are compulsory. On the other hand, some patterns might not have any dependencies or parameters.

9.1.2 Design pattern primitives and their representation

An important part of a design pattern is an explicit description of the robot behaviours that the pattern represents. A visual description, i.e., a diagram, is often very useful when a design pattern needs to be understood quickly. A textual description, that follows a well-specified syntax, is more suitable when a design pattern needs to be translated into program code.

A visual and a textual description both require a well-specified modelling language with an unambiguous syntax and semantics, so that relevant entities and processes can be expressed clearly (Harel and Rumpe, 2004). Various modelling languages for object-oriented design patterns exist, for example UML (Object Management Group, 2015), LePUS (Eden, 2011), Layout Object Model (Bosch, 1994) or DisCo (Mikkonen, 1998). While these languages are useful in traditional software development, their utility for modelling multi-agent embodied systems, such as robot swarms, is limited for two main reasons. Firstly, data in these languages is not defined explicitly, making it difficult to express where information is stored or how operations are done on it. As it has been demonstrated by the ICR framework, information flow is as important as robot behaviour when it comes to understanding and designing robot control algorithms, meaning that an adequate representation of data is required. Secondly, swarm control algorithms often rely on cooperation or communication between robots, meaning that a way of representing relationships between behaviours and data of two different robots is needed, in order to account for mechanisms that lead to emergence of desired macro-level outcomes.

Therefore, due to the shortcomings of the existing modelling languages, a new, *Behaviour-Data Relations Modelling Language* (BDRML) is proposed here. Inspired by other modelling languages such as UML and DisCo, BDRML defines a set of *primitives*, that represent behaviours and data, a set of relationships, i.e., *relations*, between these primitives and a set of *operations* that are allowed on the primitives. All these language elements have their visual as well as textual representations in BDRML.

There are three types of primitives (Figure 9.2):

- ***Behaviour***, i.e., a set of processes that deal with a particular situation a robot finds itself in, for example “Scout” or “Rest”
- ***Internal data***, i.e., information that represents a particular aspect of the environment or a robot’s internal state and that is stored in the robot’s memory
- ***External data***, i.e. information that is stored by a non-robot entity in the environment, for example by an RFID tag or by the presence of a chemical substance in an ant-inspired swarm

Note that "behaviours" in BDRML, such as "work" or "scout", can refer to "states" or sets of "states" in finite state machines. In neural network controllers, "behaviours" would not be programmed explicitly, but would manifest through the network dynamics.

Also note a crucial difference between internal and external data. Internal data is readily available to a robot at any given point in time, while external data has to be found in the environment. Moreover, when information between robots needs to be exchanged, data stored internally can only be passed from one robot to another when they meet. On the other hand, external data can be deposited by one robot into the environment and read by another robot later.

Since both behaviours and data are primitives, BDRML allows the relations between robot actions and information to be formulated. There are six types of relations possible (Figure 9.3):

- ***Transition:*** a behaviour-behaviour relation, where the robot transitions from one behavioural mode to another
- ***Read:*** a behaviour-data relation, where *internal* data, stored in the robot's memory, is used by the robot when it is engaged in a particular behavioural mode
- ***Write:*** a behaviour-data relation, where *internal* data is stored into the robot's memory when it is engaged in a particular behavioural mode
- ***Receive:*** a behaviour-data relation, where *external* data, stored in the environment, is used by the robot when it is engaged in a particular behavioural mode

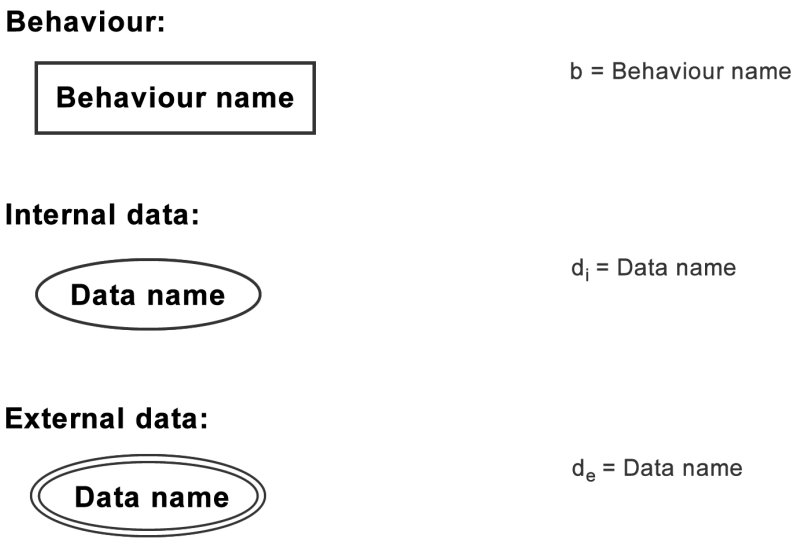


Figure 9.2: Visual (left) and textual (right) representation of the BDRML primitives.

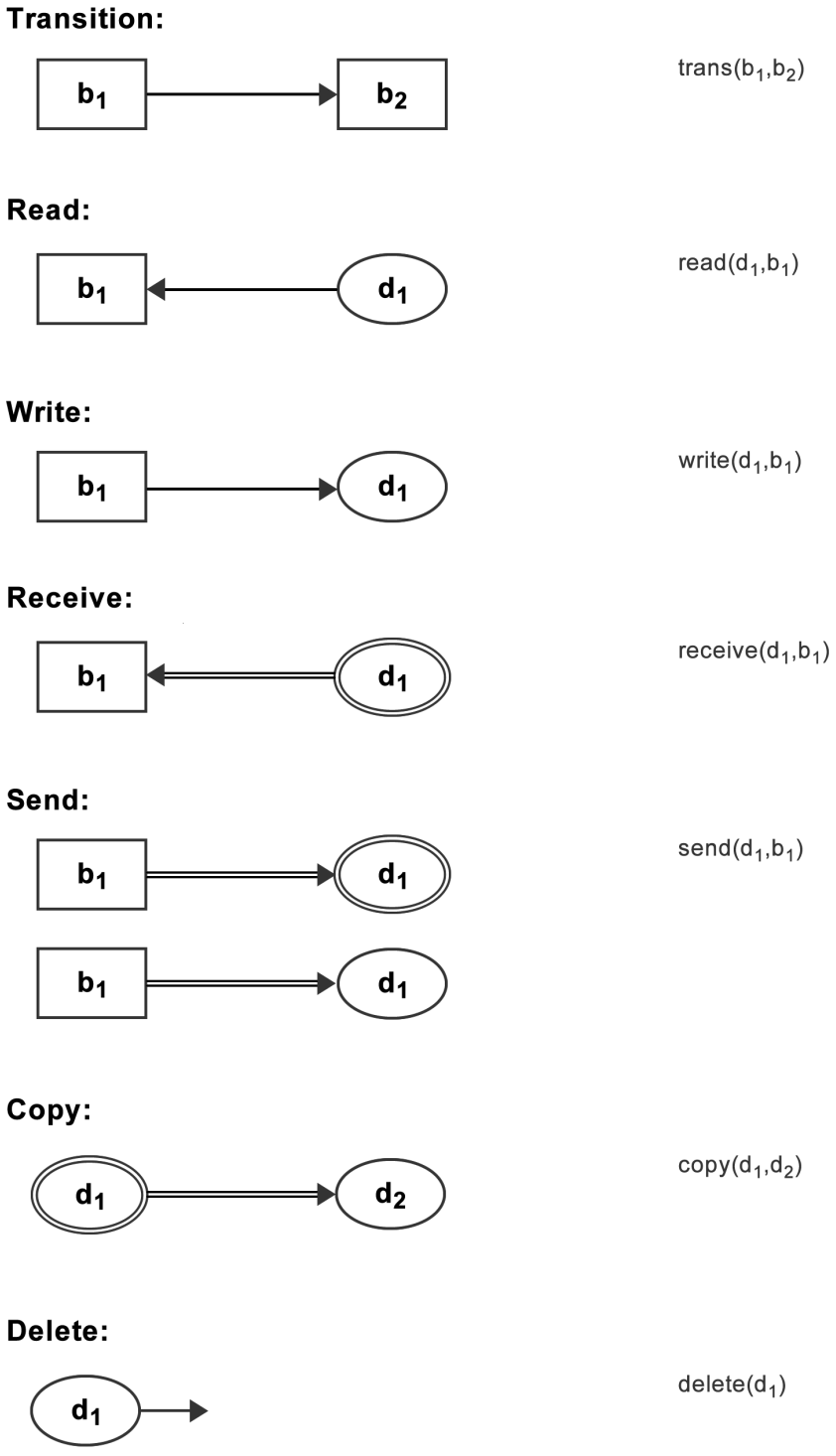
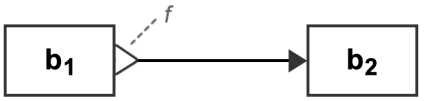


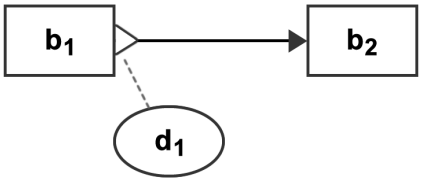
Figure 9.3: Visual (left) and textual (right) representation of the BDRML relations and operations.

A function as a condition:

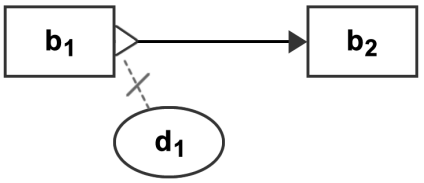


trans(b_1,b_2) : {*f*}

Existence and non-existence of data as a condition:

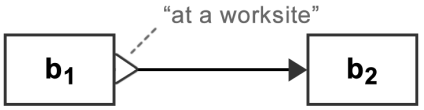


trans(b_1,b_2) : { $\exists d_1$}



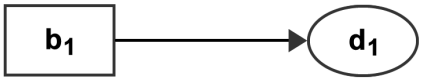
trans(b_1,b_2) : { $\nexists d_1$}

A textual description as a condition:



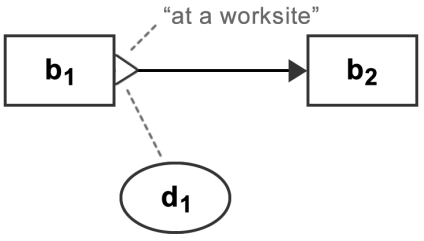
trans(b_1,b_2) : {"at a worksite"}

"Always" condition:



write(d_1,b_1) : {*}

A combination of conditions:



trans(b_1,b_2) : {"at a worksite", $\exists d_1$}

Figure 9.4: Visual (left) and textual (right) representation of the BDRML conditions.

- ***Send:*** a behaviour-data relation, where *external* data is stored by the robot into the environment when it is engaged in a particular behavioural mode. Alternatively, *internal* data of another robot is written into when a robot is in a particular behavioural mode

- ***Copy:*** a data-data relation, where information is copied from one data primitive to another (for example, from an external to an internal data structure that represent the same information)

Finally, an operation is possible (Figure 9.3):

- ***Delete:*** a data operation, where data is disposed of

It is also necessary to define a set of *conditions* under which a particular relation or operation occurs. A condition is visually represented as an annotated triangle at the beginning of a relation or operation arrow. In a textual representation, a condition set follows a relation or an operation signature and is separated from it by a colon (Figure 9.4). A condition may be annotated as a name of a boolean function or a probability, as an existence or a non-existence of a data structure, or as a simple and unambiguous textual description. A special type of condition is an "always" condition, represented by a star (*). Visually, a relation or an operation with an "always" condition may be represented without the condition triangle symbol. Multiple conditions can affect a single relation or operation. Unless otherwise specified (see Section 9.3), the "or" logical operator is assumed when conditions are combined.

Note that there are three types of lines used in BDRML. Single solid lines represent transitions between behaviours and read/write relations between behaviours and internal data structures. Double solid lines represent some form of communication and link external data structures with behaviours (in the case of the "receive" and the "send" relations) and with internal data structures (in the case of the "copy" operation). Double solid lines can also link a behaviour with an internal data structure, i.e. during the "send" operation, signifying that a robot that is engaged in a particular behavioural mode sends information to another robot, that stores the data in its own memory. Finally, dashed lines are used for annotating relation and operation conditions.

A design pattern representation in BDRML consists of both a visual and a textual specification. A set of primitives (V) is defined, followed by a list of relations and operations on them. Each box and arrow in the visual representation must have a corresponding line in the textual representation and vice versa. An example is shown in Figure 9.5.

9.1.3 Constraints

It is important to point out that while the design patterns are not implementation-specific, by default it is assumed that:

- The swarm is homogeneous
- The robots can sense their environment locally. In particular, they can sense the presence of worksites. They can also sense the presence of other robots and of obstacles nearby and resolve collision conflicts.
- The robots have a read/write internal memory
- In the case of some design patterns, it is expected that robots are capable of communication with other entities

These requirements are satisfied by the majority of robots that are currently being used in swarm robotic experiments, such as the e-pucks (e.g., in Sarker and Dahl, 2011; Castello et al., 2016), the eSwarBots (e.g., in Couceiro et al., 2012), the kilobots (e.g., in Becker et al., 2013), the s-bots(e.g., in Groß et al., 2008; Nouyan et al., 2009) and the MarXbots (e.g., in Bonani et al., 2010; Ducatelle et al., 2014). Future extensions to design patterns (see Section 10.2) could accommodate heterogeneous swarms or swarms with other non-standard properties.

9.2 Design patterns catalogue

In this section, seven design patterns, each belonging to one of the design pattern categories, transmitter, exchange and update (see Section 9.1.1), are presented. The design

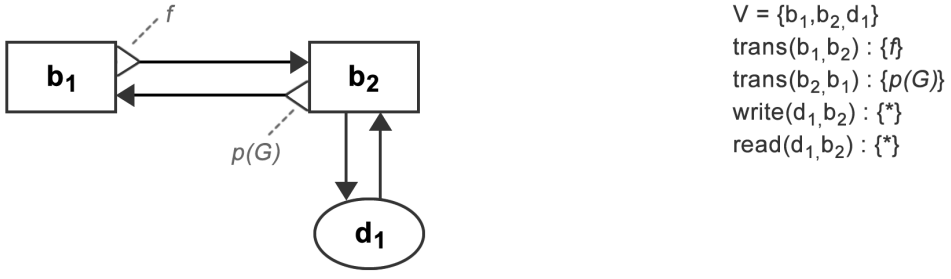


Figure 9.5: An example of a design pattern specified in BDRML. The design pattern consists of three primitives: behaviours B_1, B_2 and an internal data structure D_1. A robot changes its mode from B_1 to B_2 when a boolean function f returns true. The robot transitions back from B_2 to B_1 with a probability $p(G)$. While engaged in behaviour B_2, the robot writes to and reads from D_1.

patterns are mostly based on the control strategies used in experiments throughout Chapters 4 – 8 and relevant chapters are mentioned in the design pattern descriptions where appropriate. In some cases, the author's previously published work is also referenced.

An overview map of the design patterns, their categories, parameters and relationships, is shown in Figure 9.6.

The patterns are defined in the following format. First, the pattern's name and category is given, followed by a list of applications for which the pattern is suitable. The set of robot behaviours and data structures that the pattern represents are then described in both plain text and in BDRML. A list of the pattern's dependencies, behaviour parameters and consequences follows.

It is important to remember that a design pattern is not equivalent to a full robot control strategy. Depending on the pattern's category, only a particular aspect of robot control algorithm, such as how information is obtained and transmitted, or how it is updated, is described. The way in which multiple design patterns can be combined into robot control strategies is formalised and demonstrated in Section 9.3.

9.2.1 Individualist

Category: Transmitter pattern

Suitable applications

- Information about worksites is easily obtainable, for example when worksite density is high (see Chapters 4 and 5)
- More strongly recommended if, in addition, new information is generated in the environment over time and continuous exploration is thus important (see Chapter 6)

Behaviours and data structures

A robot scouts the environment and can find a worksite with a probability $p(F)$. Upon finding a worksite, the robot begins work and stores the information about the worksite, such as a local vector towards it, in an internal data structure ("Worksite data int." in Figure 9.7). The data structure may be updated periodically while the robot works. The robot ignores any information and actions of other members of the swarm.

Dependencies

None

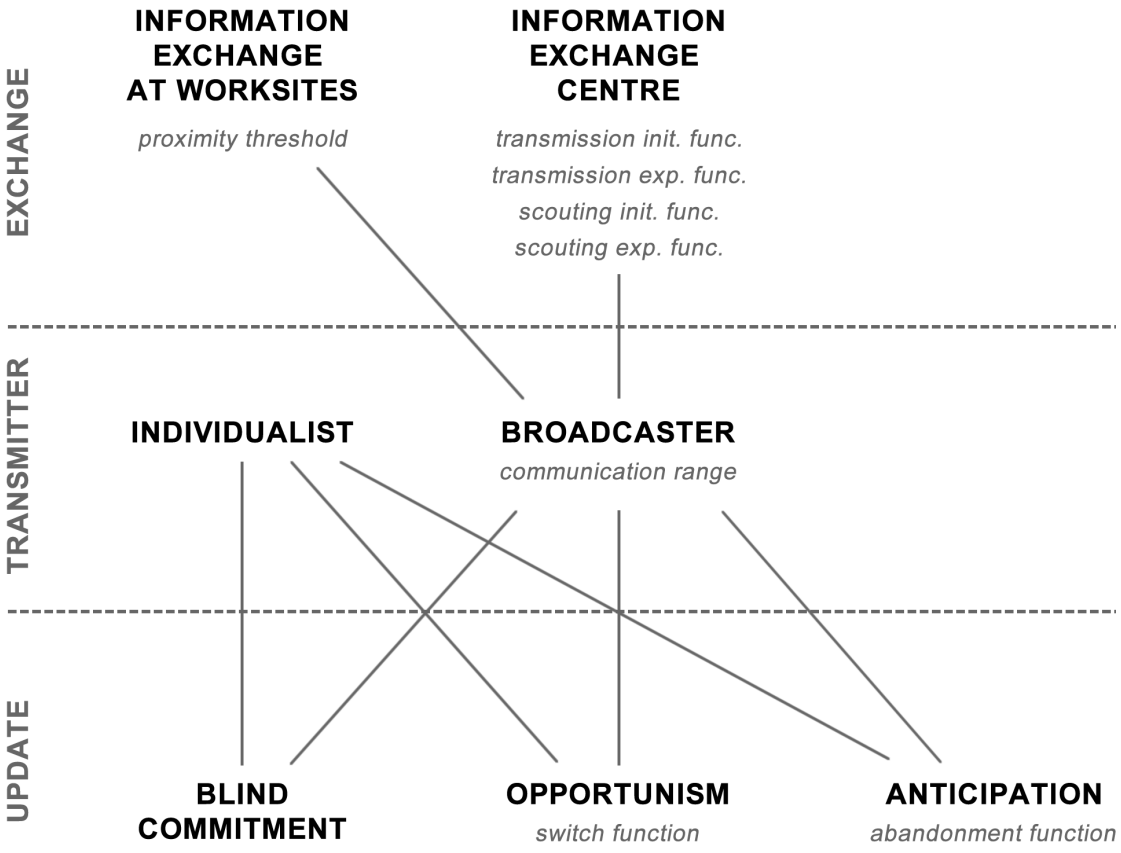


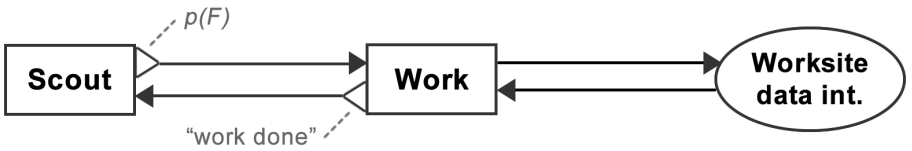
Figure 9.6: An overview map of design patterns. Design pattern categories are indicated on the left. Design pattern parameters are shown in italics below each pattern. Design patterns that can be combined together are joined by lines.

Parameters

None

Consequences

- Leads to a low information gain rate, which is the reason why information needs to be easily available to robots
- The spread of robots across worksites only depends on their movement pattern. If the movement of robots is random, an even spread across the environment is achieved.
- Prevents spread of erroneous information (Pitonakova et al., 2014)
- Minimises any misplacement and opportunity costs



V = {Scout, Work, Worksite data int.}
trans(Scout, Work) : {*p(F)*}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}

Figure 9.7: The Individualist design pattern

9.2.2 Broadcaster

Category: Transmitter pattern

Suitable applications

- Information about worksites is difficult obtain by individual robots (see Chapters 4 – 6)

Behaviours and data structures

A robot scouts the environment and can find a worksite with a probability $p(F)$. Additionally, it can receive information about a worksite from other robots while it is engaged in the "Work" behaviour. When an informed and an uninformed robot meet, they form a temporary peer-to-peer connection and the uninformed robot gets recruited to the worksite, stores information about it in its internal data structure ("Worksite data int." in Figure 9.8), and begins work. Similarly, a scout remembers and starts working on a worksite that it discovers on its own. The robot's internal data structure may be updated periodically while the robot works.

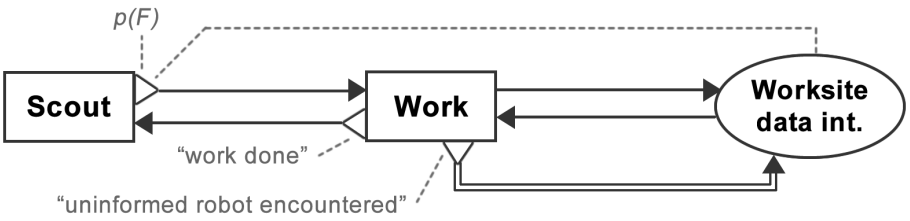
Dependencies

None

Parameters

- Robot communication range: A larger communication range causes a higher information gain rate, but also increases any misplacement and opportunity costs. Consequently, performance can decrease due to congestion and overcommitment to worksites (Pitonakova et al., 2016a).

Consequences



V = {Scout, Work, Worksite data int.}
trans(Scout, Work) : {*p(F)*, ∃ Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}
send(Worksite data int., Work) : {"uninformed robot encountered"}

Figure 9.8: The Broadcaster design pattern

- Information about worksites is more easily accessible by uninformed robots
- Information is carried and transmitted by robots, meaning that the information gain rate depends on the probability of robots meeting, i.e., on their movement algorithm and on the structure of the environment
- Causes the robots to incur misplacement cost, associated with traveling to worksites after being recruited (see Section 4.3.2)
- Increases the probability of robots to incur opportunity cost, as a result of outdated information being spread across the swarm (see Sections 4.3.3 and 5.1.4)
- Can lead to spread of erroneous information (Pitonakova et al., 2014)

9.2.3 Information exchange at worksites

Category: Exchange pattern

Suitable applications

Uninformed robots are likely to encounter information transmitters, i.e., other robots or non-robot information storage devices, near worksites, for example:

- In the consumption task, during which robots remain at worksites until they are depleted (see Chapters 4 and 6)
- When a combination of worksite density, the robot scouting strategy and the communication range of the transmitters and the uninformed robots, leads to probability of information exchange that is likely to be higher than with alternative exchange design patterns (see Chapters 4 – 6).

Behaviours and data structures

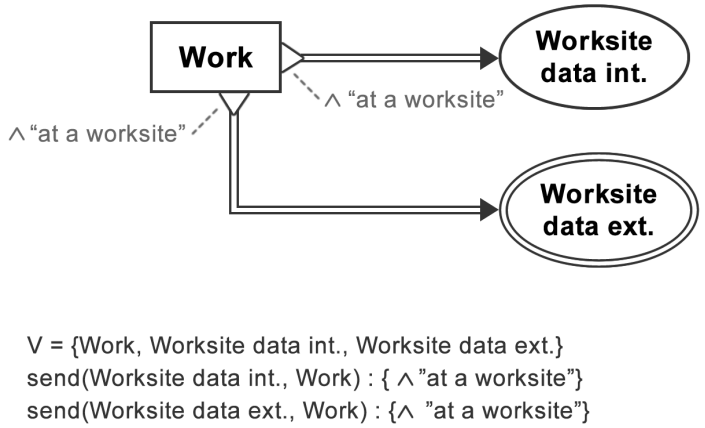


Figure 9.9: The Information Exchange at Worksites design pattern

Robots only exchange information while they are near a worksite that an informed robot is working on. Note that in the BDRML syntax, the conditions of the two relations, that connect the "Work" behaviour with the "Worksite data int." and "Worksite data ext." data structures, have an "and" operator. This notation assures that both "at a worksite" conditions always has to be met when this design pattern is combined with other patterns (see Section 9.3).

Dependencies

- The information gain rate depends on the structure of the environment and on the communication range of transmitters.

Parameters

- Proximity threshold: a maximum distance at which a robot is considered to be "at a worksite"

Consequences

- After an initial worksite discovery by a robot, the range at which other robots can find the worksite is enlarged, increasing the swarm's scouting success.

9.2.4 Information exchange centre

Category: Exchange pattern

Suitable applications

- Static and dynamic environments where worksite density is very low (see Chapters 4 – 6)
- Environments with mediocre worksite density, where the swarm task requires robots to become misplaced from worksites (e.g., the collection task), provided that the Information Exchange Centre is identical to the place where robots need to travel to periodically as a part of their task (e.g., the base, see Chapters 5 and 6)

Behaviours and data structures

Robots meet at the Information Exchange Centre (IEC) in order to exchange information. There are two types of robots found at the IEC: informed robots, that provide information and uninformed robots that search for information.

An informed robot pauses its work and returns to the IEC when its boolean *transmission initiation function*, t, returns true, in order to begin providing information at the IEC. The robot leaves the IEC based on a *transmission expiry function*, r and resumes work.

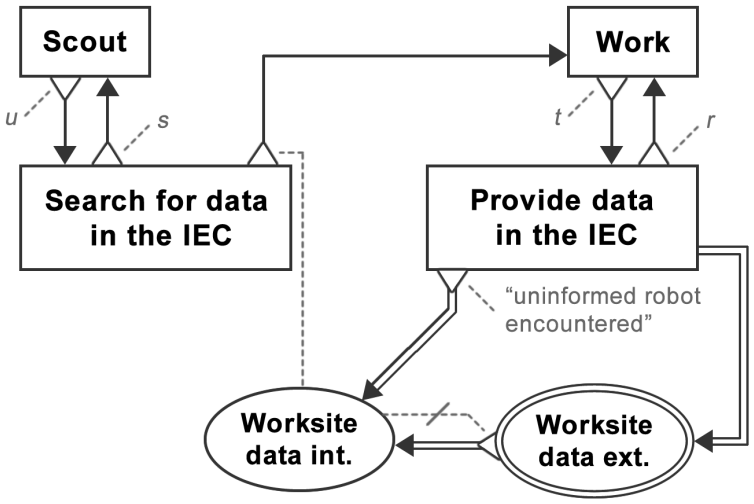
An uninformed robot located outside of the IEC, i.e., a scout, periodically returns to the IEC based on a *scouting expiry function*, u, in order to check whether new information is available. If the robot finds information about where work is located, it transitions to the "Work" behaviour and leaves the IEC. If no information is available in the IEC, the uninformed robot leaves the IEC and resumes scouting according to a *scouting initiation function*, s.

Dependencies

- The scouting efficiency of the swarm decreases due to the fact that scouts return to the IEC based on a scouting expiry function. This function must fit the nature of the environment (for example, enough time must be given to scouts to explore a large or a highly dynamic working arena), otherwise the swarm might be unable to discover worksites at all (e.g., see Section 6.1.1).

Parameters

- Transmission initiation function, t: a rule that causes informed robots to return to the IEC. For example, the need to drop off resource in the base during the collection task.
- Transmission expiry function, r: a rule that causes informed robots to leave the IEC. For example, if the IEC pattern is combined with the Broadcaster pattern, expiry of a recruitment time can trigger the robots to resume work.



V = {Scout, Search for data in the IEC, Work, Provide data in the IEC, Worksite data int., Worksite data ext.}
trans(Scout, Search for data in the IEC) : {*u*}
trans(Search for data in the IEC, Scout) : {*s*}
trans(Search for data in the IEC, Work) : { ∃ Worksite data int.}
trans(Work, Provide data in the IEC) : {*t*}
trans(Provide data in the IEC, Work) : {*r*}
send(Worksite data int., Provide data in the IEC) : {"uninformed robot encountered"}
send(Worksite data ext., Provide data in the IEC) : {*}
copy(Worksite data ext., Worksite data int.) : { ∄ Worksite data int.}

Figure 9.10: The Information Exchange Centre design pattern

- Scouting expiry function, u: a rule that causes scouts to return to the IEC. For example, expiry of a maximum scouting time.
- Scouting initiation function, s: a rule that causes uninformed robots in the IEC to become scouts. For example, robots might become scouts with a certain scouting probability each second.

Consequences

- Information gain rate is less dependent on the structure of the environment, on the communication range of robots and on the robot movement algorithm. The variance in information gain rate is small across different environment types.
- Promotes spatio-temporal coordination between robots. This is advantageous when a single worksite exists in a static or a dynamic environment. On the other hand, the swarm performance is poor when the swarm needs to concentrate on multiple worksites simultaneously.

- Incurs high misplacement and opportunity costs, relatively to the other exchange patterns. The amount of the incurred costs depends on the structure of the environment, especially on the worksite distance from the IEC. A larger worksite distance generally leads to higher costs.

9.2.5 Blind commitment

Category: Update pattern

Suitable applications

- Static environments
- Environments with unknown dynamics

Behaviours and data structures

V = {Worksite data int.}

Figure 9.11: The Blind Commitment design pattern

A robot continues to work from a worksite that it discovers and does not abandon the worksite until it is depleted.

Dependencies

None

Parameters

None

Consequences

- The swarm behaviour is relatively easy to design and predict
- Opportunities for a better swarm performance might be missed

9.2.6 Opportunism

Category: Update pattern

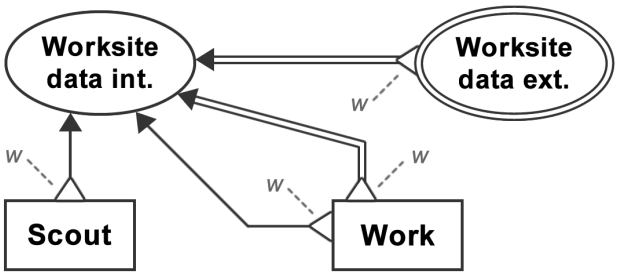
Suitable applications

Environments where it is important to extract reward from worksites with the highest utility:

- Static environments, where there is a time limit on how long a swarm can work
- Dynamic environments

Behaviours and data structures

A robot continuously evaluates the utility of its worksite and compares it to the utilities of all other worksites that it can obtain information about (as a result of scouting, communication with other robots, or discovery of information in external data storage devices). The robot abandons its own worksite and subscribes to a new one when a boolean switch function, w, returns true.



V = {Worksite data int., Worksite data ext., Scout, Work}
write(Worksite data int., Scout) : {*w*}
write(Worksite data int., Work) : {*w*}
send(Worksite data int., Work) : {*w*}
copy(Worksite data ext., Worksite data int.) : {*w*}

Figure 9.12: The Opportunism design pattern

Dependencies

- Unregulated information exchange can lead to a high increase of the misplacement and opportunity costs incurred by the robots, as well as poor sampling of the environment. For example, if Opportunism is combined with the IEC pattern, that promotes a high information gain rate, overcommitment of the majority of the swarm to a single worksite can occur, significantly decreasing the swarm performance (see Chapter 7 and (Pitonakova et al., 2016a))
- It is recommended to combine the Opportunism pattern with a transmitter or an exchange pattern where information flow is regulated to some extent, for example, the Information Exchange at Worksites pattern.

Parameters

- Switch function, w: a rule according to which a robot switches from working on a worksite W_1 with a certain utility U_1 to a worksite W_2 with a "better" utility U_2. For example, a real-value threshold can specify by how much U_2 should be greater than U_1.

Consequences

- Promotes preferential exploitation of high-reward worksites.
- Requires sampling of the utility of all worksites that a robot encounters while working. This can imply additional energy costs to the robot.

- If Opportunism is combined with another design pattern that involves communication between robots, additional data packets about worksite utilities need to be exchanged during communication. In addition, more frequent communication between robots is required, as worksite utilities of other members of the swarm need to be evaluated whenever possible. This can imply additional energy costs as well as data error accumulation and propagation (see Section 7.1.3).

9.2.7 Anticipation

Category: Update pattern

Suitable applications

Environments where it is important to extract reward from worksites with the highest utility:

- Static environments, where there is a time limit on how long a swarm has to work
- Dynamic environments

Behaviours and data structures

A robot continuously evaluates the utility of its worksite and abandons the worksite when a boolean abandonment function, a, returns true.

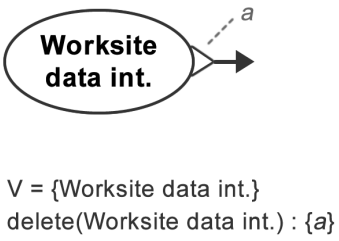


Figure 9.13: The Anticipation design pattern

Dependencies

- Worksite abandonment leads to information loss and thus to a higher amount of uncertainty cost paid by the swarm (see Section 8.2). Therefore, robots that abandon worksites must be able to discover new information relatively quickly.
- It is recommended to combine the Anticipation design pattern with another pattern that leads to a high information gain rate, e.g., the Information Exchange Centre pattern.

Parameters

- Abandonment function, a: a rule according to which a robot abandons its worksite. For example, the worksite might be abandoned when its utility falls under a specified threshold.

Consequences

- Promotes frequent sampling of the environment
- Prevents overcommitment to worksites and congestion (see Chapter 8 and (Pitonakova et al., 2016a))

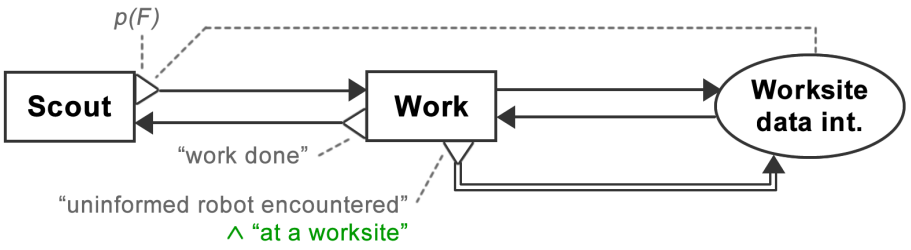
9.3 Combining design patterns

An important property of design patterns is that they can be combined together into a specific robot control algorithm, i.e., *a control strategy*. A BDRML representation of a control strategy can be created by following six design pattern combination rules:

1. Copy all sets of behaviours, B, from all patterns into a new behaviour set, B', i.e., $B' = \{B_1 \cup B_2 \cup \ldots \cup B_n\}$.
2. Copy common data structures from design pattern data structure sets D into a new data structure set, D', i.e., $D' = \{D_1 \cap D_2 \cap \ldots \cap D_n\}$.
3. Add additional data structures, that have a *read* relation with a behaviour, and do not already belong to the set D', into D'. This allows one pattern to *extend* information processing routines of another.
4. Copy all relations between the primitives that belong to sets B' or D', including their conditions. Unless it is otherwise specified by a relation condition, assume the "or" operator when combining conditions.
5. Copy all operations on the primitives that belong to sets B' or D', including their conditions. Unless it is otherwise specified by an operation condition, assume the "or" operator when combining conditions.
6. Delete all relations that belong to shorter *relation paths* between behaviours and data structures (but not between behaviours). A relation path specifies a set of relations that lead from a primitive V_1 to a primitive V_2, including those relations that pass through other primitives and create an indirect relation between V_1 and V_2. If multiple relation paths exists between a behaviour and a data structure

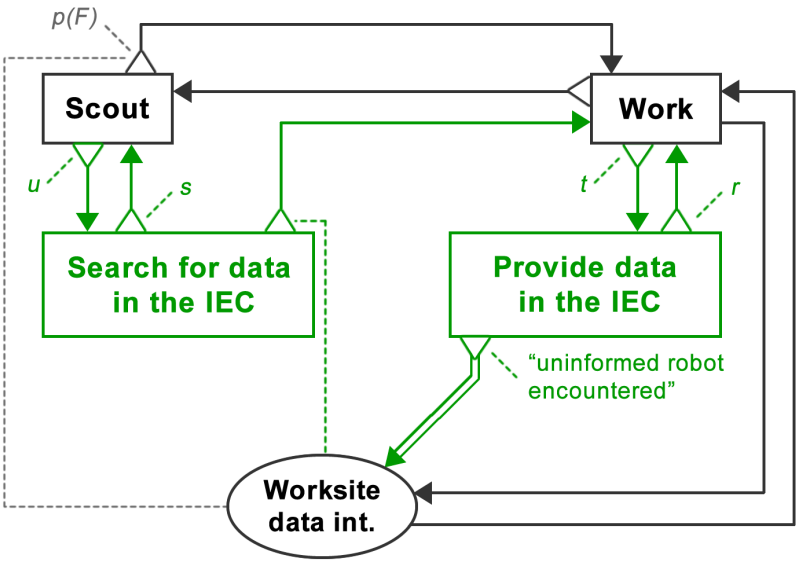
after multiple design patterns have been combined, removing relations that belong to shorter relation paths allows for one design pattern to *redefine* communication routines of another. For example, imagine that a direct *write* relation between "Work" and "Worksite data" exists in design pattern P_1. Another design pattern, P_2, defines that there is a *transition* between the "Work" and a "Stay home" behaviour and a *write* relation between "Stay home" and "Worksite data", but that "Work" and "Worksite data" are not directly related. When P_1 and P_2 are combined, the relation between "Work" and "Worksite data" should be deleted, so that P_2 can redefine the communication routine suggested by P_1. An example is shown in Figure 9.15.

Figure 9.14 shows an example of how two design patterns, Broadcaster and Information Exchange at Worksites, can be combined to create the "local broadcaster" control strategy described in Section 3.1.3. First, a set of behaviours that belong to both patterns is found. This set includes the "Scout" and the "Work" behaviours. Next, the "Worksite data int." data structure, that belongs to both patterns, is included in the control strategy. The "Worksite data ext." data structure and its relations are not copied, since the data structure does not have a *read* relation with any behaviour. The relations between all primitives that belong to the control strategy, as well as their conditions, are also included. The conditions of the relation between "Work" and "Worksite data int" are combined using the "and" operator, as specified by the Information Exchange at Worksite pattern.



V = {Scout, Work, Worksite data int., ~~Worksite data ext.~~}
trans(Scout, Work) : {*p(F)*, ∃ Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}
send(Worksite data int., Work) : {"uninformed robot encountered, ∧"at a worksite"}
~~send(Worksite data ext., Work) : { ∧"at a worksite"}~~

Figure 9.14: The "local broadcaster" control strategy, resulting from combining the Broadcaster and the Information Exchange at Worksites design patterns. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives and relations, drawn from the Information Exchange at Worksites pattern, are shown in green. Primitives and relations that were not copied are shown as strikethrough text, but they are not shown visually.



V = {Scout, Search for data in the IEC, Work, Provide data in the IEC, Worksite data int., ~~Worksite data ext.~~}
trans(Scout, Work) : {*p(F)*, ∃Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}
~~send(Worksite data int., Work) : {"uninformed robot encountered}~~
trans(Scout, Search for data in the IEC) : {*u*}
trans(Search for data in the IEC, Scout) : {*s*}
trans(Search for data in the IEC, Work) : { ∃Worksite data int.}
trans(Work, Provide data in the IEC) : {*t*}
trans(Provide data in the IEC, Work) : {*r*}
send(Worksite data int., Provide data in the IEC) : {"uninformed robot encountered"}
~~send(Worksite data ext., Provide data in the IEC) : {*}~~
~~copy(Worksite data ext., Worksite data int.) : { ∄Worksite data int.}~~

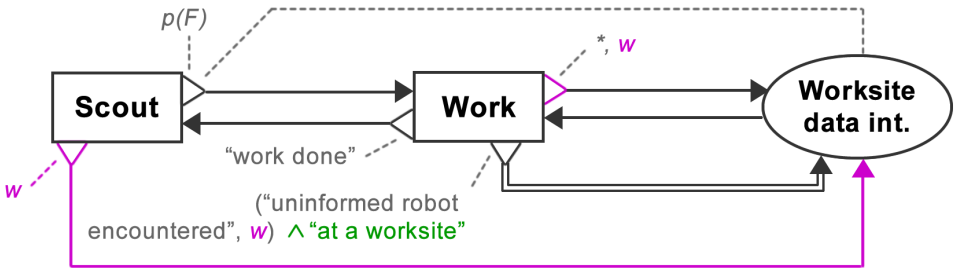
Figure 9.15: The "bee swarm" control strategy, resulting from combining the Broadcaster and the Information Exchange Centre design patterns. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives and relations, drawn from the Information Exchange Centre pattern, are shown in green. Primitives and relations that were not copied or that were deleted are shown as strikethrough text, but they are not shown visually.

A combination of the Broadcaster and the Information Exchange Centre design patterns, that results in the "bee swarm" control strategy (Section 3.1.4) is shown in Figure 9.15. The control strategy has four behaviours and one data structure, "Worksite data int.". Similarly as it was the case with the "local broadcaster" control strategy, "Worksite data ext." is not copied from the Information Exchange Centre pattern, since it does not have a *read* relation to any behaviour. The *send* relation between "Work" and "Worksite data int.", defined in the Broadcaster pattern, is deleted, since a longer relation path that passes through the "Provide data in the IEC" behaviour, inherited from the Information

Exchange Centre pattern, exists.

Three design patterns can be combined by following the same combination rules. Figure 9.16 shows how the Broadcaster, Information Exchange at Worksites and the Opportunism patterns form a "local broadcaster with opportunism" control strategy, explored in Chapter 7. The "Scout", "Work" and "Worksite data int." primitives are shared among the patterns and are a part of the control strategy. The *write* relations between "Worksite data int." and the two behaviours inherit conditions from all three design patterns. Note the condition that belongs to the *read* relation between "Work" and "Worksite data int": *, w. According to the Broadcaster pattern, the "Worksite data int." is continuously (*) updated while the robot is working. Additionally, the Opportunism pattern suggests that a *new* worksite should be adopted while a robot is in the "Work" behaviour and it finds a "better" worksite, based on the switching function, w.

Apart from a BDRML representation of the robot behaviour, other characteristics of design patterns should be considered together when design patterns are combined. The



V = {Scout, Work, Worksite data int., ~~Worksite data ext.~~}
trans(Scout, Work) : {*p(F)*, ∃ Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*, *w*}
read(Worksite data int., Work) : {*}
send(Worksite data int., Work) : {"uninformed robot encountered, ∧"at a worksite", *w*}
~~send(Worksite data ext., Work) : {∧"at worksite"}~~
write(Worksite data int., Scout) : {*w*}
~~copy(Worksite data ext., Worksite data int.) : {*w*}~~

Figure 9.16: The "local broadcaster with opportunism" control strategy, resulting from combining the Broadcaster, the Information Exchange at Worksites and the Opportunism design patterns. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives and relations, drawn from the Information Exchange at Worksites pattern, are shown in green. Additions drawn from the Opportunism pattern are shown in magenta. Primitives and relations that were not copied are shown as strikethrough text, but they are not shown visually.

list of suitable applications becomes more specific when multiple patterns form a control strategy. Or, from an developer's point of view, a more detailed specification of the swarm's environment and task allows for a higher number of design patterns to be combined together with a greater confidence. For example, if a swarm is required to collect easy-to-find rubbish from a city square but has no specific constraints for its operation, the design patterns catalogue suggests to implement a robot control strategy by combining the Individualist and the Blind Commitment patterns. On the other hand, an application may be more specific, for example a swarm may be required to collect minerals that are difficult to find and appear in mineral veins of varying richness, while the robots can only operate when enough sunshine is provided for their solar batteries. In this case, the design patterns catalogue would suggest to combine the Broadcaster, the Information Exchange Centre and the Opportunism patterns.

Secondly, the list of control strategy parameters also grows when multiple patterns are combined. In order to avoid creating a robot control algorithm with a large parameter space that needs extensive optimisation, it is advisable to prefer simpler control strategies that are based on as small number of design patterns as possible. Similarly, design patterns with a smaller number of parameters should be preferred, unless there is a reason for using a more complicated pattern. The problem with parameters is that they require decisions to be made by robot designers when the swarm is being built. Unless an exhaustive list of situations is considered during the optimisation phase, or unless a suitable on-line parameter learning algorithm is implemented, each new parameter can lead to undesirable results. For example, the Information Exchange Centre pattern requires a parameter to be set for the "scouting expiry function", in order to specify when a scout should return back to the IEC. Setting this parameter to an inappropriate value can prevent the swarm from discovering worksites, when not enough time is given to scouts to explore the environment, or from communication between robots to take place, when scouts spend too much time outside of the base and do not meet with recruiters in the base (see Section 6.1.1 and Appendix A, Section A.3.2).

9.4 Discussion

Before discussing how the design patterns proposed here have been used in the existing literature, it is important to consider the level at which the design patterns describe robot behaviour, as well as the design pattern categorisation. It has been suggested that design patterns for swarm robotics should describe multiple *levels* of behaviour. For example, "local-level" or "basic" patterns should describe how robots interact, while "global-level" or "composed" patterns should describe swarm-level behaviour, such as "labour division" (Nagpal, 2004; Gardelli et al., 2007; Fernandez-Marquez et al., 2013). On the contrary, all design patterns presented here describe the same *level* of robot

behaviour, equivalent to the "local-level primitives" of (Nagpal, 2004) or the "basic design patterns" of (Fernandez-Marquez et al., 2013). The *control strategies*, for example, "solitary swarm with anticipation" represent a combination of design patterns, and are similar to the "global-level primitives" of (Nagpal, 2004) or the "high-level patterns" of (Fernandez-Marquez et al., 2013). The control strategies, however, are not design patterns themselves. Instead, they are particular design pattern realisations in swarm applications that fit specific mission requirements.

It is proposed here that an information- and cost-based description of individual, "local-level", robot behaviour is an appropriate level at which design pattern should be defined. A detailed, lower-level description, that deals with a particular object-oriented or functional implementation on a robot would have to include details about a particular experiment or particular robot hardware that would potentially not be useful to a developer with slightly different hardware or application. Similarly, description of macroscopic, "global-level", swarm behaviour, for example a "flocking pattern" (Fernandez-Marquez et al., 2013), would simply be a re-description of a combination of robot behaviours that fit a particular swarm mission. Such global-level description would also potentially contain a lot of parameters. On the other hand, describing parts of robot behaviour that deal with particular problems, without providing too many implementation details, allows for modularity and reusability.

While they describe the same level of behaviour, the design patterns presented here are categorised based on what particular aspect of robot behaviour they represent, with respect to obtaining, sharing and updating of robot information. They thus follow the categorisation methodology of object-oriented design patterns (Gamma et al., 1994) and multi-agent systems design patterns (Aridor and Lange, 1998). Each design pattern belongs to one of the three categories: transmitter, exchange and update.

Various combinations of these patterns can be found throughout the literature. The Individualist and the Blind Commitment patterns are often used when simple foraging algorithms are needed as a basis for robot behaviour, while other swarm behaviours, such as self-regulation or task-allocation are explored (e.g., Krieger and Billeter, 2000; Campo and Dorigo, 2007; Labella et al., 2006).

The Individualist pattern is also often used when performance of swarms without and with communication is compared (e.g., in Balch and Arkin, 1994; Rybski et al., 2007; Fujisawa et al., 2014; Gutiérrez et al., 2010; Lee et al., 2013). It is usually the case that swarms in such experiments forage from deposits that are difficult to find, resulting in better performance of swarms that utilise communication.

A combination of the Broadcaster and Information Exchange at Worksites patterns has been explored for example in (Sugawara and Watanabe, 2002). Confirming the design pattern characteristics presented here, the authors showed that increasing the strength

of interaction between robots (e.g., as a result of a large swarm size or a large communication range of robots) leads to performance improvement, but that the improvement is sub-linear. In other words, in the ICR framework terminology, a high information gain rate often leads to high misplacement and opportunity costs that prevent the performance from improving linearly with the amount of information that the robots can get. Similarly, (Valdastri et al., 2006) argued that recruitment in swarms that used the Broadcaster and the Information Exchange at Worksites patterns leads to increased congestion (i.e., a higher misplacement cost), as well as propagation of old information through the swarm (i.e., a higher opportunity cost).

In the work of (Rybski et al., 2007), swarms that used a combination of the Broadcaster and Information Exchange at Worksites patterns did not outperform swarms that used the Individualist pattern. The authors proposed that the relatively poor performance of swarms that utilised communication was a result of communication noise. However, the characteristics of the Broadcaster design pattern point to two additional possible explanations. Firstly, only four robots were used in the experiments, and it is possible that they did not meet often enough for communication to make a positive difference to their performance. Secondly, the robots were collecting pucks that were fairly far apart from one another, with respect to the size of the robot body. A robot that was recruited to a puck thus had to perform search in order to locate another puck nearby. In such a setup, it is possible that the negative effect of misplacement and opportunity costs outweighed the positive effect of recruitment.

The Broadcaster pattern has also been combined with the Information Exchange Centre pattern, often in bee inspired robot swarms and agent-based simulations. In these experiments, robots collect items from the environment and return them to the base, where they also recruit in a peer-to-peer fashion. Since the Information Exchange Centre pattern is the most suitable when items of interest are difficult to find but need to be collected into a central place, swarms that use it outperform other, non-communicating, swarms in foraging experiments where items of interest are clumped in a small number of patches (Krieger and Billeter, 2000; Bailis et al., 2010; Dornhaus et al., 2006; Lemmens et al., 2008; Thenius et al., 2008). Interestingly, contrary to the characteristics of both of these patterns, Dornhaus et al. and Lemmens et al. did not find any negative effects of communication, such as the increase of congestion or fast depletion of resource deposits, in their simulations. They thus claimed that the swarm performance increases linearly with swarm size. A closer inspection of their algorithm reveals that their agents were allowed to occupy the same space, meaning that the physical aspect of agents was not fully modelled, preventing misplacement cost from being incurred as a result of congestion. Furthermore, opportunity cost could also not be incurred in their simulations, since resource deposits had unlimited volumes.

Robot control algorithms that contain the update patterns can also be found throughout

the literature. The Opportunism pattern has been used in foraging simulation experiments where agents preferred to head towards resource deposits that were closer to the base, i.e., deposits that allowed a faster collection of resource (Lemmens et al., 2008; Gutiérrez et al., 2010). In accordance with the results presented in this thesis, Gutiérrez et al. showed that Opportunism causes the majority of a swarm to concentrate on a single resource deposit when the information spread in the swarm is not regulated.

Opportunistic behaviour, where a swarm prefers to forage from more profitable resource deposits, can also be achieved when robots that utilise the Information Exchange Centre pattern and recruit in the base, recruit for a longer amount of time when their deposits are more profitable (Schmickl et al., 2012). Even though the mechanism of achieving opportunism is different than when unemployed robots simply prefer to be recruited to better deposits, the results of such behaviour are similar. In line with the Opportunism design pattern characteristics, Schmickl et al. demonstrated that a sufficient amount of scouting in a swarm where robots behave opportunistically is very important for the ability of the swarm to appropriately react to environmental changes.

Finally, the Anticipation design pattern has been used in a control algorithm that allowed robots to decide which type of puck they should search for in order to maintain a desired density of puck types in a drop off location (Jones and Mataric, 2003). The abandonment function, that caused a robot to stop foraging for a particular puck type, was related to locally perceived behaviour of other members of the swarm.

Design patterns allow us to consider a broad range of experiments with different robot hardware and identify building blocks of robot behaviour that fit specific swarm mission requirements. Other design methodologies exist, for example probabilistic finite state machine models (e.g., Liu and Winfield, 2010; Mather and Hsieh, 2012; Reina et al., 2014) and evolutionary algorithms (e.g., Niv et al., 2002; Sperati et al., 2011; Ferrante et al., 2015; Doncieux et al., 2015). Unlike design patterns, these methodologies are more suitable for parameter optimisation, rather than for behaviour selection. Therefore, design patterns compliment these methodologies when it comes to developing robot control algorithms. See Section 10.2 for a further discussion.

9.5 Extending the design patterns catalogue

The transmitter design patterns presented above either did not make use of communication or relied on local, peer-to-peer, communication between robots. Another type of communication, called *stigmergy*, involves the exchange of information between agents through the environment. Algorithms that utilise stigmergy are often inspired by pheromone-based communication, characteristic for ant colonies. In order to help their nest mates to navigate towards food, ants leave chemicals called *pheromones* on the ground while

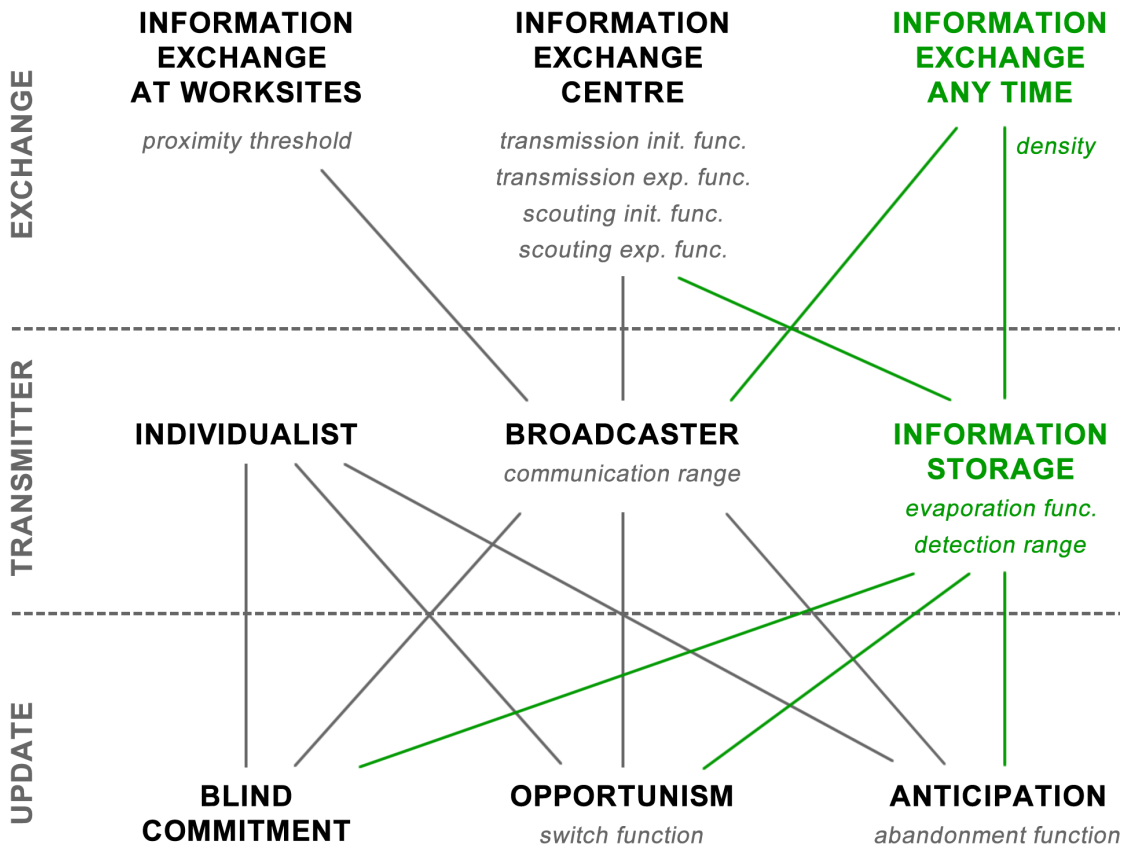


Figure 9.17: An extended overview map of design patterns, with the new patterns shown in green. Design pattern categories are indicated on the left. Design pattern parameters are shown in italics below each pattern. Design patterns that can be combined together are joined by lines and any extra parameters required for the pattern combination are shown next to the lines.

traveling back and forth between the nest and the food, forming trails in the environment (Beekman, 2001; Arab et al., 2012). Other ants can sense pheromones and thus use the pheromone trails in order to navigate during foraging. Pheromones evaporate in a certain rate, assuring that a path that is no longer being used, for example because the food source has been depleted, eventually disappears and does not recruit more workers.

There are two aspects of stigmergy that are interesting from the design pattern perspective. Firstly, it is the involvement of a medium that is stationary and external to the robots and that holds information relevant to the work that the swarm performs. Secondly, it is the fact that information is available in many locations across the work arena, rather than only being exchanged in the base or near worksites. Two design patterns can be created in order to capture these aspects of robot behaviour. A transmitter pattern, called Information Storage, according to which information is stored in

data storage devices, and an exchange pattern, Information Exchange Any Time, according to which information can be exchanged anywhere in the work arena. In this section, these two patterns are formalised using information from experiments found in the swarm robotics literature. Their description is not as detailed as that of the patterns presented in Section 9.2, since no experiments that could thoroughly test the suggested robot behaviours have been performed yet. Nevertheless, it is demonstrated here that the new design patterns can easily be combined with the other design patterns according to the design pattern combination rules defined in Section 9.3. Figure 9.17 shows the new patterns on an extended design pattern map.

9.5.1 Information storage

Category: Transmitter pattern

Suitable applications

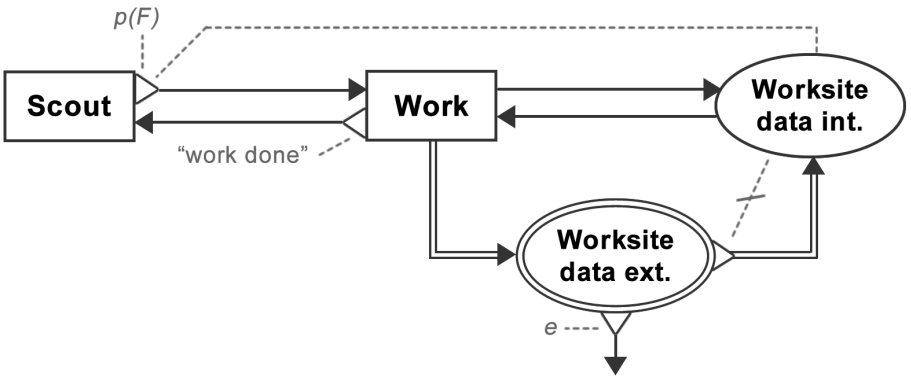
- Information about worksites is difficult to obtain by individual robots (Balch and Arkin, 1994; Fujisawa et al., 2014)

Behaviours and data structures

A robot scouts the environment and can find a worksite with a probability $p(F)$. Additionally, it can receive information about a worksite if it finds a *data storage device* ("Worksite data ext." in Figure 9.18) located in the environment. Once a robot discovers information about a worksite, either as a result of scouting or when finding a data storage device, it stores data about the worksite in its memory ("Worksite data int." in Figure 9.18) and begins work. The robot's internal data structure is updated periodically while the robot works.

An informed robot stores information about its worksite into data storage device(s) when appropriate. For example, when the design pattern is combined with the Information Exchange Any Time pattern (see below), special data storage devices, such as RFID tags, may be dropped into the environment and updated by the robot (e.g., in Drogoul and Ferber, 1993; Hrolenok et al., 2010). Chemicals that mimic ant pheromones, such as alcohol, can also be used (e.g., in Mayet et al., 2010; Fujisawa et al., 2014). Alternatively, stationary robots, that do not directly participate in work, can be used to store information (e.g., in Ducatelle et al., 2011; Hoff et al., 2013). On the other hand, when the Information Storage and the Information Exchange Centre design patterns are combined, information is stored in a central location, for example in the robot base.

The information is deleted from the storage device(s) according to an *evaporation function*, e.



V = {Scout, Work, Worksite data int., Worksite data ext.}
trans(Scout, Work) : {*p(F)*, ∃ Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}
send(Worksite data ext., Work) : {*}
copy(Worksite data ext., Worksite data int.) : { ∄ Worksite data int.}
delete(Worksite data ext.) : {*e*}

Figure 9.18: The Information Storage design pattern

Dependencies

- The evaporation function affects how long the information about worksites remains available in each storage device, i.e., the *lifespan* of the stored information. The function must consider dynamics of the environment. If the information life span is too long, robots follow information to depleted worksites and incur a high opportunity cost. On the other hand, a very short information life span prevents robots finding and utilising the stored information (Drogoul and Ferber, 1993).

Parameters

- Evaporation function, e: a rule according to which information in the data storage device(s) is deleted or considered as too old. This function plays a similar role as the *evaporation rate* of ant pheromones. For example, information might have a pre-defined lifespan length. Upon lifespan expiration, the storage device deletes the information, if such an ability has been programmed into it. Alternatively, robots that read the information also evaluate its age and decide whether it should be considered as too old to use.
- Detection range: a range at which a robot can find a storage device.

Consequences

- Information about worksites is more easily accessible by uninformed robots
- Information is stored in the environment, meaning that the information gain rate depends on the probability of robots detecting the information storage devices, but not on the probability of robots meeting each other
- Causes robots to incur misplacement and opportunity costs, as a result of recruitment to remote worksites. The extent of these costs increases with an increasing swarm size, as a result of congestion (Drogoul and Ferber, 1993; Parker, 1995; Hoff et al., 2010). The amount of costs paid may however be smaller than when Information Exchange Centre pattern is used, since data storage devices may be located closer to worksites.

9.5.2 Information exchange any time

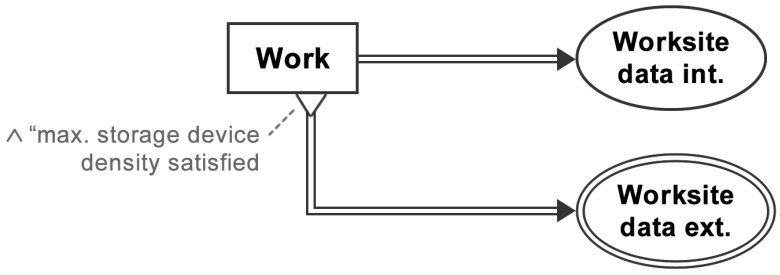
Category: Exchange pattern

Suitable applications

Unknown

Behaviours and data structures

Information is exchanged anywhere in the environment.



V = {Work, Worksite data int., Worksite data ext.}
send(Worksite data int., Work) : {*}
send(Worksite data ext., Work) : {∧ "max. storage device density satisfied"}

Figure 9.19: The Information Exchange Any Time design pattern

Dependencies

Unknown

Parameters

- Maximum storage device density: When the Information Exchange Any Time pattern is combined with the Information Storage pattern, the maximum allowed

information storage device density must be specified in order to prevent the environment from being cluttered with storage devices. For example, the minimum distance between two RFID tags should be specified. In case of chemical trails, this could be related to the frequency at which the chemical is deposited into the environment.

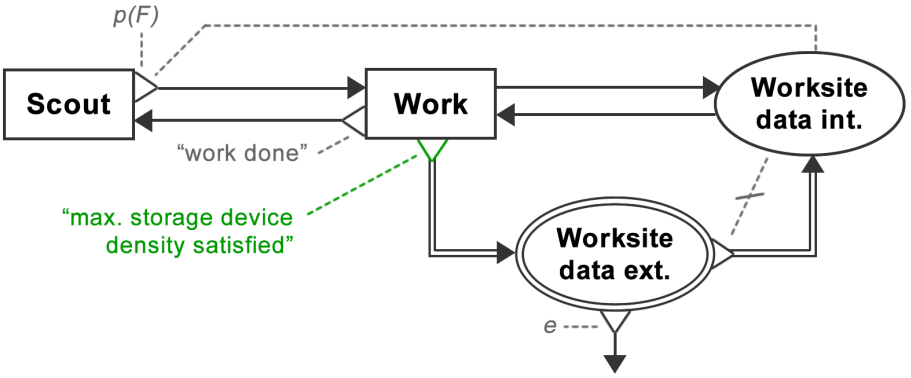
Consequences

Unknown

9.5.3 The new design pattern combinations

The new design patterns can be combined together as well as with other patterns from the design patterns catalogue by following the design pattern combination rules defined in Section 9.3.

Using the Information Storage and the Information Exchange Any Time patterns together results in an ant-inspired robot control strategy, depicted in Figure 9.20, where

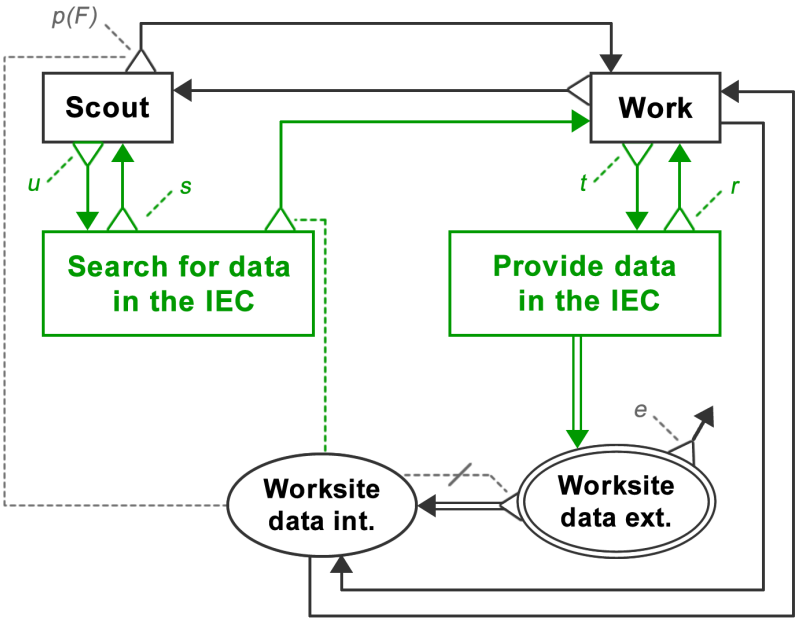


V = {Scout, Work, Worksite data int., Worksite data ext.}
trans(Scout, Work) : {*p(F)*, ∃ Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}
send(Worksite data ext., Work) : {"max. storage device density satisfied"}
copy(Worksite data ext., Worksite data int.) : { ∄ Worksite data int.}
delete(Worksite data ext.) : {*e*}
~~send(Worksite data int., Work) : {*}~~

Figure 9.20: A control strategy resulting from combining the Information Storage and the Information Exchange Any Time design patterns. Primitives, relations and operations of the Information Storage pattern are shown in black. Additional relations, drawn from the Information Exchange Any Time pattern, are shown in green. Relations that were deleted are shown as strikethrough text, but they are not shown visually.

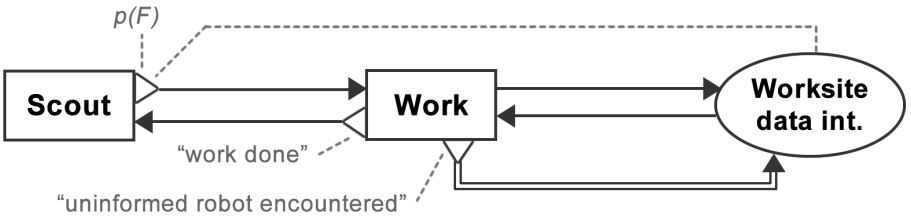
robots utilise stigmergy and can find information about worksites in the environment (e.g., in Drogoul and Ferber, 1993; Parker, 1995; Hoff et al., 2010; Fujisawa et al., 2014).

Robots designed by combining the Information Storage and the Information Exchange Centre patterns (Figure 9.21) store information about where worksites are located in the base (e.g., in Alers et al., 2011). Unsuccessful scouts arrive to the base in order to read the information and begin work.



V = {Scout, Search for data in the IEC, Work, Provide data in the IEC, Worksite data int., Worksite data ext.}
trans(Scout, Work) : {*p(F)*, ∃Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}
~~send(Worksite data ext., Work) : {*}~~
copy(Worksite data ext., Worksite data int.) : { ∄Worksite data int.}
delete(Worksite data ext.) : {*e*}
trans(Scout, Search for data in the IEC) : {*u*}
trans(Search for data in the IEC, Scout) : {*s*}
trans(Search for data in the IEC, Work) : { ∃Worksite data int.}
trans(Work, Provide data in the IEC) : {*t*}
trans(Provide data in the IEC, Work) : {*r*}
~~send(Worksite data int., Provide data in the IEC) : {"uninformed robot encountered"}~~
send(Worksite data ext., Provide data in the IEC) : {*}

Figure 9.21: A control strategy resulting from combining the Information Storage and the Information Exchange Centre design patterns. Primitives, relations and operations of the Information Storage pattern are shown in black. Additional primitives and relations, drawn from the Information Exchange Centre pattern, are shown in green. Relations that were deleted are shown as strikethrough text, but they are not shown visually.



V = {Scout, Work, Worksite data int., ~~Worksite data ext.~~}
trans(Scout, Work) : {*p(F)*, ∃ Worksite data int.}
trans(Work, Scout) : {"work done"}
write(Worksite data int., Work) : {*}
read(Worksite data int., Work) : {*}
send(Worksite data int., Work) : {"uninformed robot encountered"}
~~send(Worksite data ext., Work) : { ∧ "min. storage device density satisfied"}~~

Figure 9.22: A control strategy resulting from combining the Broadcaster and the Information Exchange Any Time design patterns. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives and relations, drawn from the Information Exchange Any Time pattern, are shown in green. Primitives and relations that were deleted are shown as strikethrough text, but they are not shown visually.

The Information Exchange Any Time pattern can also be combined with the Broadcaster pattern (Figure 9.22), leading to behaviour where robots exchange information when they meet anywhere in the work arena (e.g., in Balch and Arkin, 1994; Gutiérrez et al., 2010; Fraga et al., 2011; Ducatelle et al., 2014).

9.6 Summary

A design pattern represents a particular aspect of robot behaviour that addresses a specific swarm mission requirement, such as finding worksites given a certain worksite density, dealing with certain environmental dynamics, etc. A design pattern is created by considering results of experiments with a particular robot behaviour, for example a bee-inspired information exchange in a central "base" location, and by generalising knowledge learned during the experiments using the Information-Cost-Reward framework.

Each design pattern presented here belongs to one of the three categories: transmitter, exchange and update. Transmitter patterns identify entities that should transmit information. For example, a transmitter pattern might suggest that robots should share information among each other, or that robots should exchange information via RFID tags placed in the environment. Exchange patterns dictate where information should be exchanged, for example whether robots can communicate any time when they meet each other, or whether a specific meeting place should be designated. Update patterns deal

with how individual robots update their information, for example whether they continuously search for "better" worksites or whether they decide to abandon and forget their worksites when certain conditions are satisfied.

A description of a design pattern includes its name, category, a list of suitable applications, description of robot behaviours including their parameters, dependencies on other behaviours of the robot, and the consequences of the design pattern on the swarm's scouting efficiency, information gain rate and tendency to incur the misplacement and the opportunity costs. The description of robot behaviours is provided using the Behaviour-Data Relations Modelling Language, BDRML. A BDRML-based description consists of visual and textual representation of robot behaviours and data structures used by the behaviours, as well as conditional relations and operations between and on them.

Using BDRML, multiple design patterns can be unambiguously combined into a control strategy by following the design pattern combination rules. The design patterns catalogue introduced here provides robot designers with knowledge about suitable applications, parameters and consequences of various robot behaviours and thus allows them to devise suitable robot control algorithms based on known mission characteristics.

Chapter 10

Conclusion

This chapter first summarises the motivation, methodology and outcomes of the work presented in this thesis. The summary is followed in Section 10.2 by a discussion of the design patterns as a methodology for swarm robotics design. Advantages and disadvantages of this approach are addressed, and relevant future research directions are identified. The chapter concludes with final remarks on how the ICR framework for understanding swarm behaviour developed in this thesis can help us to gain an insight into swarm intelligence.

10.1 Thesis summary

10.1.1 The challenges in swarm robotics

Demand for autonomous multi-robot systems, where robots can cooperate with each other without human intervention, is set to grow rapidly in the next decade. Today, technologies such as self-driving cars and fleets of robotic assistants in hospitals and warehouses are being developed and used. In the future, autonomous robot swarms could be deployed in retrieval, delivery, reconnaissance and construction missions on Earth and on other planets.

A swarm of robots as a whole can exhibit complex behaviour, such as selection of the most profitable worksites, or self-regulation of work force, without the need for a "supervisor" in the loop. Compared to traditional robotic applications, where a single robot performs all the work, or where multiple robots are controlled by a centralised controller, distributed collective systems have desirable properties such as low cost of individual robots, robustness, fault tolerance and scalability (Brooks, 1989; Mataric et al., 2003; Campbell and Wu, 2011; Fernandez-Marquez et al., 2013). However, unlike traditional robotics, swarm robotics requires a "bottom-up" approach to behaviour design

(Trianni et al., 2011; Parunak and Brueckner, 2015). While performance of the swarm is specified and evaluated on the macro, collective, level of the swarm, robot designers need to program the control algorithms of individual robots, which requires them to take into account complex robot-robot interactions that underpin the emergence of collective intelligence. In order to be able to develop such systems, we need a methodology that aligns bottom-up design decisions with top-down design specifications. While many experiments have been performed with various robot swarms (most recently, e.g., Ducatelle et al., 2014; Fujisawa et al., 2014; Paull et al., 2014; Hoff et al., 2013; Kernbach et al., 2013), and some effort has also been put into learning how these systems can be designed (e.g., De Wolf and Holvoet, 2007; Hamann, 2013; Brambilla et al., 2014; Reina et al., 2015), a framework that organises and explains experimental results is still missing. It is currently unclear which design decisions can be generalized to which tasks or why swarms designed in a particular way exhibit the behavior that they do.

In this thesis, a novel approach for understanding and designing robot swarms has been proposed. Based on thousands of different simulation experiments, the Information-Cost-Reward (ICR) framework has been formulated. This framework relates the way in which a swarm obtains and uses information to suitability of its control strategy in a particular mission. The ICR framework has been applied to identify and describe design patterns for robot swarms. Design patterns are modular aspects of robot behaviour that define when and how information should be obtained, exchanged or updated by robots, given particular mission characteristics. Multiple design patterns can be combined together in order to create a suitable robot control strategy.

10.1.2 The experiments and the framework for understanding results

In this thesis, robot swarm behaviour has been studied in the context of collective *consumption* and *collection* tasks. In the consumption task, robots search for work in the environment and obtain reward by staying close to worksites while gradually depleting them. In the collection task, robots extract small packets of resource from worksites and deliver them to the base. These tasks model a number of real-world swarm-robotic missions, such as cleaning of streets, collection of raw materials, picking up and delivering packages, or maintaining a shop floor.

Three basic robot control strategies, *solitary*, *local broadcaster* and *bee*, have been used to create three types of homogeneous robot swarm. Robots in solitary swarms search the environment and perform work on their own, and do not share information with each other. Local broadcasters help each other to find worksites by broadcasting recruitment signals while performing work. Bee swarm robots meet in the base in order to exchange information about where worksites are located. These strategies have been analysed and compared in a number of *static* and *dynamic* environments with different worksite distributions and worksite volumes.

The experimental results have been explained in context of the Information-Cost-Reward (ICR) framework, formulated in Section 4.4.5. Under this framework, a swarm is understood as a decentralised cognitive entity that perceives the environment, acts on information that it obtains and changes the environment as a result of its actions. Scout robots serve as the swarm's *sensors*, as they search the environment for worksites, i.e., for information about where work needs to be done. They do so with a certain *scouting efficiency*, approximated by how quickly the swarm can make the first worksite discovery (Section 4.2.1). Upon finding worksites, scouts become workers, i.e. *actuators* of the swarm, and they utilise information about where work is located in order to obtain reward. Successful scouts can also use some form of communication in order to recruit other members of the swarm to work. The rate at which information flows into the swarm (via scouting) and through the swarm (via communication) is quantified as the *information gain rate* (Section 4.2.3).

Depending on the type of control strategy used, workers might need to pay certain *costs* before they can obtain reward. *Misplacement cost* (Section 4.3.2) is incurred by informed robots that are some distance away from their worksites, for example because they have to drop off resource in the base, because they were recruited at a location that is some distance from the worksite, or because they cannot reach the worksite due to congestion. *Opportunity cost* (Section 4.3.3) is incurred by robots that are subscribed to worksites that have been depleted and therefore cannot yield any more reward. While attempting to reach their worksites, such robots loose the opportunity to discover active, non-depleted, worksites or to be recruited to them.

The swarm inevitably changes its environment as a result of robots performing work. The swarm's scouting success may decrease over time as worksites get depleted, unless new worksites appear in the environment. Furthermore, if the task or the control strategy requires robots to move away from their worksites, for example to drop off resources or to recruit in the base, the tendency for robots to incur opportunity cost may increases over time.

The four swarm characteristics (scouting efficiency, information gain rate and the tendency to incur misplacement and opportunity costs) affect how suitable a control strategy is in a given environment. Furthermore, there are four environmental characteristics that are important when selecting a suitable control strategy. *Worksite density* determines how probable it is that worksites can be found. *Worksite volume* affects how quickly a worksite gets depleted once robots start working on it, i.e., the trade-off between exploitation of a single worksite by multiple robots and exploration of the environment. *Misplacement of reward from worksites* is also important. When reward is misplaced, robots need to travel away from worksites in order to obtain reward, for example as is the case in the collection task, where robots need to drop off resource in the base. Finally, the *dynamics of the environment* determine whether the worksite characteristics, such as their location or utility, change over time. In static environments, worksites do not

change, unless the swarm performs work on them. In dynamic environments, worksite characteristics are transient and affected by factors external to the swarm, meaning that reward needs to be extracted from them as quickly as possible.

Control strategies with a high information gain rate, such as the local broadcaster and the bee strategies, are suitable when worksite density is low and when worksites are difficult to find, especially in static environments when worksites have large volumes. However, since high information gain rate is usually achieved by communication between robots, tendency of swarms to incur misplacement and opportunity costs is higher when recruitment is utilised. Swarms with a high information gain rate thus usually have lower performance than swarms with a low information gain rate in less difficult environments, where worksites are easy to discover (see Chapters 4 and 5). Furthermore, when the environment is dynamic, a good scouting efficiency and a low tendency to incur costs are important, as new information is periodically generated in the environment and needs to be discovered by the swarm (see Chapter 6). Therefore, bee swarms, that have a low scouting efficiency and usually pay a high amount of misplacement and opportunity costs, are outperformed by local broadcaster or solitary swarms in most dynamic environments. Finally, costs can be ameliorated by the nature of the swarm's task. For example, the negative effect of misplacement cost during the collection task is smaller in bee swarms, since bee swarm robots recruit in the same place where resource needs to be dropped off (see Chapter 5).

Apart from the three basic control strategies, two biologically inspired add-on strategies, *Opportunism* (Chapter 7) and *Anticipation* (Chapter 8), have also been explored. Each add-on strategy refined the behaviour of a basic control strategy in order to find out whether the swarm's performance improved in dynamic environments. In particular, the add-on strategies addressed the two following problems that the swarms faced in dynamic environments: Firstly, worksites are temporary and as much reward needs to be extracted as quickly as possible. Secondly, new worksites appear in the environment over time, meaning that the work arena needs to be periodically scouted for new information. The Opportunism add-on control strategy caused robots to subscribe to worksites with utilities higher than that of previously encountered worksites, causing the swarm to preferentially work in higher-reward areas. The Anticipation add-on strategy allowed a swarm to scout the environment more frequently by causing robots to abandon worksites that were likely to be depleted soon.

The experiments revealed a complex interplay between the add-on strategies, the basic control strategies and the environmental characteristics. Opportunism positively affected local broadcasters in many scenarios, as it allowed the robots to exploit more profitable and less crowded worksites. On the other hand, it was not suitable to combine with the bee control strategy, where robot interactions were frequent and information could spread very quickly. Opportunism caused the majority of a bee swarm to concentrate on a single worksite, creating congestion and preventing the swarm from exploring

the environment. The Anticipation behaviour led to information loss, as a result of worksite abandonment. It was therefore only suitable to combine with the bee control strategy during the collection task, where robots could visit the base in order to quickly obtain new information. Anticipation significantly deteriorated the performance of solitary and local broadcaster swarms, where the robots could not discover new information quickly enough.

10.1.3 Design patterns for robot swarms

By using the ICR framework, we can understand swarm performance on a relatively abstract level and thus obtain general knowledge from specific experiments. The framework describes robot behaviour in terms of "information flow" instead of "control code" and the results of the behaviour in terms of individual "costs" and "rewards" rather than simply "performance improvement" or "performance degradation". Therefore, behaviour description that uses the terminology of the ICR framework is not dependent on particular robot hardware or software and the results of the behaviour can be understood in a detailed manner and in relation to the characteristics of the environment that the swarm operates in. Most importantly, such general understanding allows us to identify modular aspects of robot behaviour that can addresses specific swarm mission requirements, such as finding rare worksites, or dealing with certain environmental dynamics, and formalise them as design patterns (see Section 9.1). Multiple design patterns can then be reliably and unambiguously combined into *a control strategy* by following the design pattern combination rules (Section 9.3).

Each design pattern presented in this thesis belongs to one of three categories: *transmitter*, *exchange* and *update*. Transmitter patterns identify entities, such as robots or RFID tags, that should store and transmit information. Exchange patterns determine where information should be exchanged, for example whether robots can communicate any time, or whether a specific place for information exchange should be designated. Update patterns determine how individual robots update their own information, for example whether they remain subscribed to their worksites until they deplete them, or whether they continuously search for "better" worksites to switch to.

The following design patterns have been identified:

- **Individualist:** a transmitter pattern, according to which robots do not share information with each other (Section 9.2.1)
- **Broadcaster:** a transmitter pattern, according to which robots form temporary peer-to-peer connections in order to communicate (Section 9.2.2)

- **Information Storage:** a transmitter pattern, where robots utilise external information storage devices, such as RFID tags, in order to store and share information (Section 9.5.1)
- **Information Exchange at Worksites:** an exchange pattern that restricts information sharing to when robots are near worksites (Section 9.2.3)
- **Information Exchange Centre:** an exchange pattern that requires robots to meet in a single designated place in order to exchange information (Section 9.2.4)
- **Information Exchange Any Time:** an exchange pattern that allows robots to exchange information at any point during their work cycle (Section 9.5.2)
- **Blind Commitment:** an update pattern, according to which robots remain subscribed to their worksites until the worksites are depleted (Section 9.2.5)
- **Opportunism:** an update pattern, according to which robots evaluate worksites around them and switch to more profitable worksites whenever possible (Section 9.2.6)
- **Anticipation:** an update pattern, according to which robots evaluate worksites that they are working on and abandon them if a certain criterion is specified (Section 9.2.7)

Description of a design pattern includes its name, category, a list of suitable applications, description of robot behaviours including their parameters, a set of dependencies on other behaviours of the robot, and a list of consequences of the design pattern on emergent swarm characteristics. Description of robot behaviours is formalised using the *Behaviour-Data Relations Modelling Language*, BDRML (Section 9.1.2). A BDRML-based description consists of visual and textual representation of robot behaviours and data structures used by the behaviours, as well as conditional relations and operations between and on them. Data structures can be internal, i.e., stored in a robot's memory, or external, i.e., stored in the environment, for example in RFID tags.

BDRML allows the creator of a design pattern to unambiguously define a relevant part of a robot control algorithm, without the need to specify programing code implementation for particular robot hardware. Unlike in other modeling languages, such as UML (Object Management Group, 2015), LePUS (Eden, 2011), Layout Object Model (Bosch, 1994) or DisCo (Mikkonen, 1998), the relationships between robot behaviours and data structures are expressed explicitly in BDRML, including relationships between behaviours of one robot and data of another. This allows BDRML to easily represent control algorithms where robots cooperate and share information with each other.

Robot designers can refer to the design patterns catalogue (Section 9.2 and 9.5) when they know at least some of the swarm mission parameters and want to identify what

type of robot behaviour they should implement in order to best address the mission requirements. For instance, if a swarm is required to collect easy-to-find rubbish from a city square but has no specific constraints for its operation, the design patterns catalogue suggests to combine the Individualist and the Blind Commitment patterns into a fairly simple robot control strategy. When the mission details are more demanding and specific, for example mineral collection from difficult-to-find mineral veins of varying richness, while the robots can only operate when enough sunshine is provided for their solar batteries, more complex strategies can be devised. In this case, combining the Broadcaster, the Information Exchange Centre and the Opportunism patterns would be recommended.

10.2 Discussion and future work

The role of swarm design patterns is to provide guidelines about how to achieve suitable collective (macro) behaviour by implementing robot (micro) control algorithms. It has been recognised for some time that in order to be able to design robot swarms, we need a general understanding of swarm intelligence that does not rely on implementation details but rather captures something abstract about how emergent collective behaviour works and how it can be controlled (e.g., Parunak and Brueckner, 2004; Serugendo et al., 2006; Winfield, 2009; Brambilla et al., 2014). Similarly to the solution proposed in this thesis, many experts believe that an information-based understanding can help us to generalise experimental results (e.g., Moussaid et al., 2009; Wang et al., 2012; Polani et al., 2013; Miller et al., 2014), and that we should pay attention to feedback loops between the swarm and the environment (e.g., Gardelli et al., 2007; Fernandez-Marquez et al., 2013). It has also been proposed that description of robot behaviour should be modularised, since properties and interfaces of smaller behavioural components can be defined more easily (e.g., Brazier et al., 2002; Nagpal, 2004; Parunak and Brueckner, 2015).

Various mathematical modeling approaches have been used in order to show how micro behaviour can be optimised to deliver desired macro results, for example "property-driven design" (Brambilla et al., 2014), "swarm calculus" (Hamann, 2013) or other methods based on probabilistic finite state machines (e.g., Liu and Winfield, 2010; Mather and Hsieh, 2012; Reina et al., 2014). A common approach is to define a function for evaluating whether a swarm's performance is satisfactory (e.g., "collect 60% of resource in 1 hour") or optimal (based on the optimal foraging theory, see Section 2.2), and then to devise a set of differential equations that describe how the robot population changes between possible robot "states" (e.g., "scouting" and "foraging"). State transitions are probabilistic and subject to parameters. Parameters that the robot designer has no control over, such as the effect of interference between robots, are estimated or measured by performing targeted experiments. Other parameters, such as recruitment time, can be controlled and optimised in order to fit a desired outcome. Once a set of differential

equations has been correctly parametrised, predictions about swarm performance can be made by calculating results in new environments, without the need to perform more experiments or to run computationaly expensive agent-based simulations.

The first problem of this approach is that the set of differential equations is often too specific for a particular application or particular robot hardware. Furthermore, unlike design patterns, mathematical models describe the system as a whole, which means that they are often complex and challenging to communicate about. It is therefore difficult to learn anything general from these models and to share knowledge among different research groups. On the other hand, design patterns are modular and implementation non-specific, which makes them easier to understand, reuse and cooperate on. They do not dictate what values parameters should take, but rather provide guidelines on how parameter values affect the macro behaviour of the system.

A more serious problem with mathematical modeling is that in order for it to be useful, there already has to be an algorithm for robot behaviour selected by the algorithm developer. There are no guidelines on whether the algorithm should include communication, how the robots should scout, where they should communicate, etc. Such decisions are often made somewhat arbitrarily, based on the developer's prior experience with a particular hardware, or knowledge of a particular biologically inspired behaviour. While mathematical models can help with algorithm *optimisation*, design patterns guide algorithm *selection*. The role of design patterns is to offer alternative ways in which robot control algorithms can be constructed. Naturally, once an appropriate control algorithm has been selected, it can be optimised by using mathematical modeling or evolutionary algorithms.

It has been repeatedly shown throughout this thesis, as well as in the literature (e.g., Campbell and Wu, 2011; Mataric et al., 2003), that algorithm selection is not a trivial task due to the complexity of relationships between the swarm and the environment. Without design patterns, a reliable algorithm selection could only be achieved via open-ended evolution, where alternative ways of gathering, exchanging and updating information would be subject to evolutionary pressures. However, not only would such open-ended evolution be computationally very expensive, it would not help us to understand anything about swarm intelligence. The evolved strategies would work as required by our fitness functions, but we would have to do further analysis, for example by using the ICR framework, if we wanted to understand why they work.

Despite of their advantages, the swarm design patterns, as they currently stand, have a number of issues that need addressing. It is currently difficult to specify the exact mission conditions under which a design pattern is understood to be appropriate to apply or combine with other patterns. The design patterns need to be tested in a larger number of experimental scenarios, and more importantly, validated on real robots and refined. The design patterns catalogue also needs to be extended to include more swarm

tasks, for example reconnaissance or construction. Finally, it is important that the catalogue is publicly available and that cooperation between different research groups on extending and refining the design patterns is facilitated. The rest of this section addresses these challenges.

It is currently difficult to express the exact criteria under which a design pattern should be applied. For example, the Broadcaster pattern description suggests that the pattern should be used when "worksites are difficult to find". Is it possible to quantify this "difficulty"? Is there a specific threshold of worksite density that clearly distinguishes environments where this pattern is and is not suitable?

It is possible to relate environmental and desired swarm characteristics. For example, we know that a swarm's scouting strategy and its information gain rate affect how the swarm performs in environments with different worksite densities. However, it is difficult to formalise these relationships. Furthermore, since swarm design patterns describe robot behaviours that lead to emergent swarm performance, the impact of a design pattern on the swarm characteristics may change when the pattern is combined with another. It is therefore possible that we will never be able to identify, with a complete certainty, the most suitable set of design patterns by simply by browsing the design patterns catalogue. We could, however, perform more experiments with each design pattern, for example with different robot hardware and environments, to make the design pattern specifications more detailed. It is also important to remember that, by offering a list of dependencies and consequences, design patterns can identify situations in which they *should not* be used. For example, it is clear that if we use the Information Exchange Centre pattern, combining it with the Opportunism pattern would result in poor swarm performance.

Rather than thinking about design patterns as branches of a decision tree, that asks us questions about the environment and eventually leads us to one suitable solution, we should think about the design patterns catalogue as a source of suitable alternatives that a robot designer can consider. Object-oriented design patterns play a similar role in software engineering. It is up to the developer to decide which patterns to use, based on the context of the application. A number of patterns and their consequences can be considered for a given mission, and possibly implemented in simulation or on robots for comparison. Naturally, the choice of suitable design patterns is also affected by hardware that is currently available. For example, if we know that we do not have access to RFID tags or other means of storing information in the environment, we know that the Information Storage pattern cannot be used. Or, if a particular robot has a very small communication range, it might be better to use the Information Exchange Centre pattern, where communication range is not as important, compared to the Broadcaster pattern, where communication range significantly affects the success of recruitment.

There is a lot of work that can be done in order to improve the design patterns catalogue. The existing patterns need to be tested in more environments and, more importantly, on real robots, in order to validate their properties. A number of assumptions have been made in simulation, and the effect of these assumptions on the behaviour of the investigated swarms is currently unknown. In particular, there are three questions that need to be answered:

- **How does the behaviour of robots in the real world differ from that in the simulation?** A number of assumptions have been made in simulation, e.g., what the robot movement speed is, how carrying an object affects the robot's perception and movement, how quickly the robots can handle collisions, etc. These factors may or may not play an important role when it comes to evaluating suitability of particular design patterns in particular environments. It is thus important to compare swarms that use the same algorithms and the same type of robot hardware in both simulation and in the real world. This is possible with the simulation setup used for this thesis, since the experiments were run in the ARGoS simulation environment, that has been specifically designed for development and testing of control code for the e-puck and MarXbot robots (Pinciroli et al., 2012; Ducatelle et al., 2014).
- **How does noise affect navigation and communication or robots?** All the design patterns developed here make use of a number of sensors, e.g., light and wheel sensors, and of an infra-red-based communication module (see Sections 3.1.1 and 9.1.3). Reliability of these sensors under real world noise conditions, and the likelihood and impact of errors in the data that robots require to make decisions, is an important topic that can only be fully explored outside of simulation.
- **How do the robots interfere with each other?** Two types of interference between robots have been identified. *Physical interference*, where robots prevent each other from moving due to congestion, and *exploitational interference*, where robots compete for active worksites, which may lead to swarm loosing the opportunity to exploit other worksites (see Sections 4.1.2 and 4.1.3). Evaluating how these interferences manifest themselves in real-world robot swarms is important for understanding how different design patterns affect a swarm's tendency to incur misplacement (Section 4.3.2) and opportunity (Section 4.3.3) costs.

It is possible that the properties of design patterns, such as their dependencies, consequences and suitable applications, as they currently stand, will need to be refined and that new properties, undiscovered in simulation, will be added. Validating and refining the existing design patterns would provide more information to robot designers and thus make it easier to select between different alternatives when a robot control algorithm needs to be constructed.

The design patterns catalogue could be extended by performing experiments with new algorithms and by turning them into new design patterns, using the methodology described in Section 9.1. Design patterns for different swarm tasks, such as dispersion, aggregation or collective construction, could be added. It would be interesting to see whether behaviour modules originally discovered during experiments with the collection and consumption tasks would be applicable to other tasks or whether they would be possible to combine with modules discovered when working on other tasks. It is reasonable to believe that some patterns could be generalised enough to be applicable across different swarm tasks. For example, if the Broadcaster pattern was applied for robot aggregation, "work" could be understood as "move towards desired location", and a "worksite" as a "desired location". During a construction task, "work" could mean "build" and "worksite" would be equivalent to a "building site".

Further extensions of the design patterns catalogue could include generalising the existing design patterns and creating new patterns for heterogeneous robot swarms and swarms that work alongside humans. In these cases, entities that have different types of behaviour would work alongside each other. It is therefore important to remember that consequences of a design pattern on macro-level behaviour only apply to a subgroup of robots that implement the pattern. If particular consequences only manifest themselves in a homogeneous swarm, or if different macro-level behaviour emerges depending on the behaviour of other entities in the system, the description of such consequences should include the conditions under which they occur.

Environments with more complex dynamics also need to be explored. For example, it is reasonable to assume that some robot missions would involve worksites with different priorities, or worksites that could only be serviced by a subset of robots. Tasks that require cooperation between robots, for example collective transport, are also an interesting avenue of research. In other missions, the ability of robots to perform work or to navigate the environment may change over time, for example due to weather or human traffic conditions. Understanding the impact of previously unexplored environmental dynamics on the ability of robots to obtain, share and utilise information is vital for both refining and extending the design patterns catalogue.

Finally, the design patterns catalogue should be made publicly available, so that it can make a real impact on the research and application of robot swarms. It is vital that scientists and engineers can share knowledge and cooperate on making the catalogue larger and better specified. The BDRML language might need to be refined and extended, so that it is able to accommodate a broader range of robot control algorithms. Furthermore, implementation examples and code libraries could be added in order to facilitate the usage of design patterns. Object-oriented design patterns nowadays come with implementation examples in various programming languages, which makes them easier to understand and apply.

10.3 Final remarks

Robot swarms are a promising technology that could transform the way in which we manage logistics, transportation and agriculture, how we take care of our cities and of our environment, as well as how we explore and colonise new planets. One of the biggest challenges of using this technology is that it is not obvious how we can relate the robot behaviour, programmable by software developers, to a desired swarm-level outcome. Design patterns based on the ICR framework presented here have a potential to take robot swarms from carefully controlled laboratory conditions to real world applications, by providing tested design solutions to known classes of robot missions.

Apart from its practical applications, the ICR framework also has implications for the field of swarm cognition (Trianni et al., 2011). By understanding a swarm as a single entity that gathers and processes information, the framework helps us to gain insights into how collective intelligence emerges from interactions of relatively simple units. The ICR framework allows us to ask fundamental questions, such as what the importance of embodiment in collective intelligence is or what properties of information flow inside of a distributed system lead to a desired collective action.

Without a deeper understanding of collective intelligence, the best we can hope for is to mimic biological swarms by performing parameter optimisation. While that by itself is a significant achievement, since swarms found in nature are elegant problem-solvers capable of complex decision making, it stands to reason that we should strive to understand how exactly their intelligence emerges. By taking their behaviour apart, and by carefully examining its functional modules, we can devise new and unique module combinations, unseen in nature, that fit our own requirements.

Appendix A

Robot parameter optimisation

The behaviour of robots that belong to the three robot swarms introduced in Chapter 3, solitary, local broadcaster and bee, is subject to a number of parameters, listed in Table A.1. The particular parameter values were selected in order to optimise the performance of each swarm across different scenarios in the static consumption and collection tasks.

The swarm performance was measured in terms of the percentage of resource collected from the environment during 50 independent simulation runs. Each run lasted one simulated hour. In order to speed up the process of parameter optimisation, a subset of the scenarios that were explored during the main experiments in Chapters 4– 8 were selected (Table A.2). During parameter optimisation, the number of robots, $N_R = 25$, the number of worksites, $N_W = \{1, 4, 25\}$, and the worksite distance from the base, $D = \{9, 13, 17\}$m.

In the following sections, the individual robot parameters are explained and the impact of their values on performance of the three swarms is described. The neighbourhood search radius, r_N, and the neighbourhood search time, T_N, are explored separately for each swarm, since all control strategies utilise neighbourhood search (see Sections 3.1.2– 3.1.4). The scouting probability, $p(S)$, the maximum scouting time, T_S, and the recruitment time, T_R, are only relevant to the bee swarms.

Parameter description	**Symbol**	**Value**
Neighbourhood search time	T_N	180 seconds
Neighbourhood search radius	r_N	3 metres
Scouting probability (bee swarm)	$p(S)$	10^{-3}
Maximum scouting time (bee swarm)	T_S	18 minutes
Recruitment time (bee swarm)	T_R	120 seconds

Table A.1: Parameter values used by various robot control strategies described in Sections 3.1.2 – 3.1.4.

Simulation parameter	Set of values for main experiments	Set of values for parameter optimisation
Number of robots, N_R	{10, 25, 50}	{25}
Number of worksites, N_W	{1, 2, 4, 25}	{1, 4, 25}
Worksite distance from base, D	{5, 9, 13, 17, 21}m	{9, 13, 17}m

Table A.2: Simulation parameter values used during the main experiments in Chapters 4– 8 and during parameter optimisation.

A.1 Solitary swarms

A.1.1 Neighbourhood search

The neighbourhood search radius, r_N and the neighbourhood search time, T_N, control the neighbourhood search behaviour. A robot performs neighbourhood search when it arrives to a location where it believes that a worksite should be, but the worksite cannot be immediately detected. There are two reasons why a believed worksite location can be empty. The worksite could have been depleted by other members of the swarm while the robot was away from it, for example during the collection task. The robot might also arrive at a wrong location as a result of odometry errors. Performing neighbourhood search assures that minor odometry errors can be corrected. During the search, a robot travels randomly within a circle with a diameter of r_N metres around the believed worksite location for T_N seconds.

Since solitary swarms do not use recruitment, neighbourhood search is only needed during the collection task, when a robot arrives to a believed worksite location after dropping resource in the base. Figure A.1 shows the performance of solitary swarms with various value combinations of the r_N and T_N parameters. The parameters did not have a strong effect on the swarm performance, especially when worksites were close to the base. When worksites were further away from the base, i.e., when odometry errors were more likely to accumulate, the parameters mostly affected the performance of swarms in the Heap1 scenario, where it was the most difficult to discover a worksite. Solitary swarms had a low performance at both ends of the explored parameter space, i.e., when $(r_N, T_N) = (1\text{m}, 60\text{s})$ and when $(r_N, T_N) = (4\text{m}, 300\text{s})$. When small values of r_N and T_N were used, the robots could not cover a sufficiently large area and thus could not correct their odometry errors. On the other hand, robots with large values of r_N and T_N spent too much time performing the neighbourhood search, which caused them to waste a lot of time when there were no worksites nearby.

These results suggest that suitable parameter values, that lead to a good performance across different scenarios, lie in the middle of the parameter space. The values selected for solitary swarms were thus $(r_N, T_N) = (3\text{m}, 180\text{s})$.

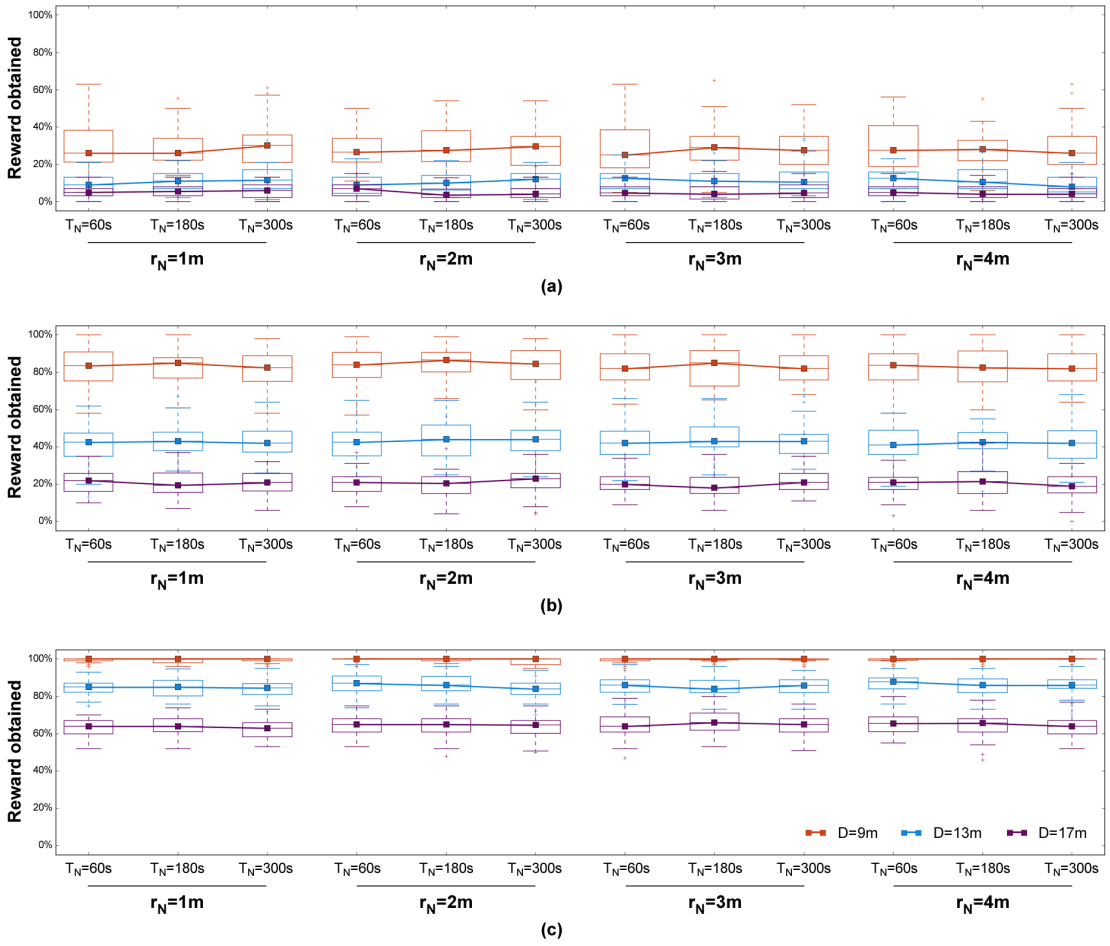


Figure A.1: The percentage of available reward obtained by 25-robot solitary swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static collection task, using various values for the neighbourhood search radius, r_N, and the neighbourhood search time, T_N, parameters. The parameter values mostly did not affect the swarm performance. The performance deteriorated in Heap1 scenarios when low values for both parameters were used.

A.2 Local broadcasters

A.2.1 Neighbourhood search

Local broadcasters make use of neighbourhood search on two possible occasions. Similarly as solitary robots, local broadcaster robots returning from the base to the worksite during the collection task might arrive to a wrong location as a result of odometry error. Furthermore, recruited robots in both the collection and the consumption task might arrive at a wrong location, since they receive information about where a worksite is from recruiters while they are some distance away from the advertised location. However, since robots are recruited near worksites, odometry errors that result from

recruitment are fairly unlikely to occur and are very small. The consumption task is thus not considered in this section.

Similarly as it was the case for solitary swarms, values of the r_N and T_N parameters did not have a strong effect on the performance of local broadcasters (Figure A.2). The most significant effect was evident in the Heap1 scenarios with large worksite distance ($D = 17$m), where extremely low or extremely large values of r_N and T_N caused the swarm performance to deteriorate.

The values $(r_N, T_N) = (3\text{m}, 180\text{s})$ were selected for local broadcasters, making the neighbourhood search behaviour consistent with that of solitary swarms.

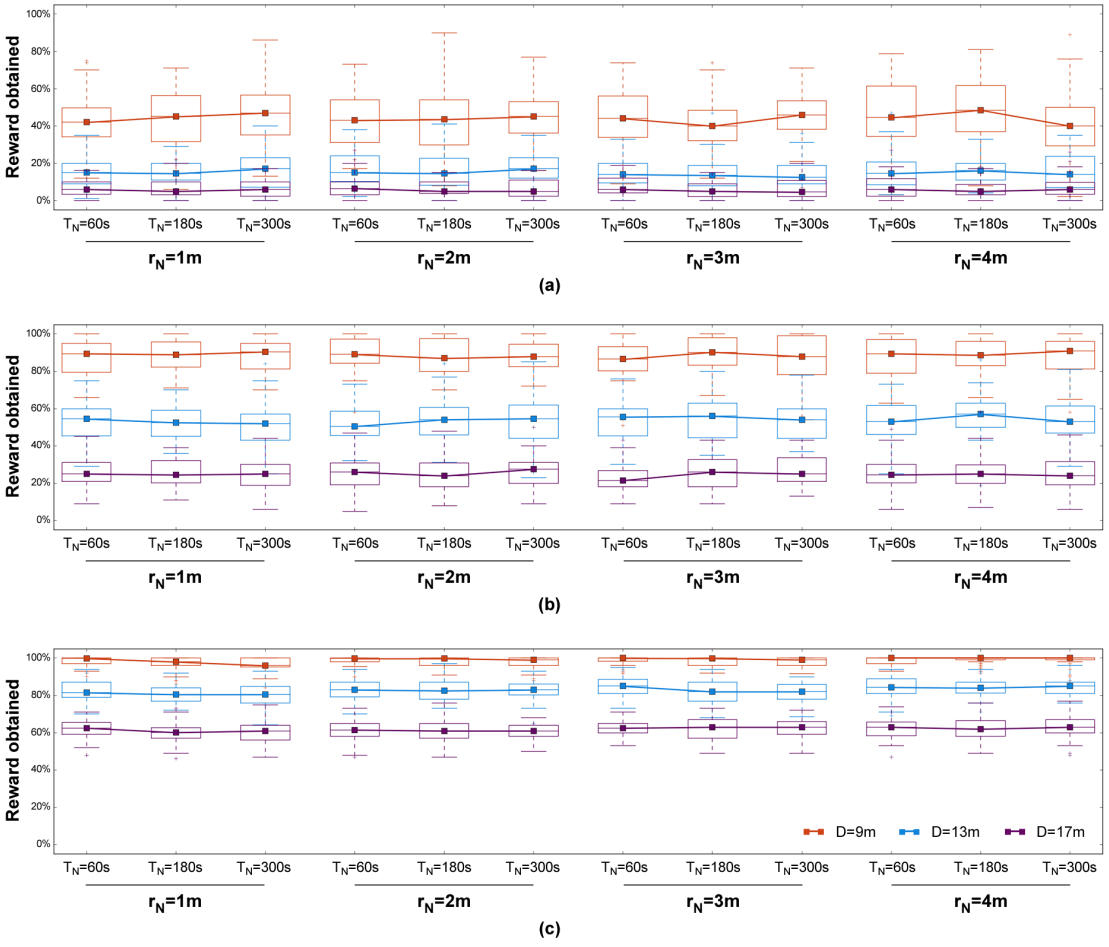


Figure A.2: The percentage of available reward obtained by 25-robot local broadcaster swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static collection task, using various values for the neighbourhood search radius, r_N, and the neighbourhood search time, T_N, parameters. The parameter values mostly did not affect the swarm performance.

A.3 Bee swarms

There were five different parameters that needed to be optimised for bee swarms: scouting probability, $p(S)$, recruitment time, T_R, maximum scouting time, T_S, neighbourhood search radius, r_N and neighbourhood search time, T_N. A suitable combination of these parameters existed in a five-dimensional parameter space that was difficult to fully explore. The parameters were thus organised into parameter groups based on the behaviour that they represented. Different value combinations of parameters from the same group were explored at the same time, while values of parameters from other groups were meanwhile set to values estimated as suitable during a quick visual inspection of the swarm behaviour. Table A.3 lists the parameter groups in the order in which they were optimised, as well as the values of parameters from other groups that were set during the optimisation.

Parameter group	**Explored parameter values**	**Set parameters**
Scouting versus recruitment (Section A.3.1)	$p(S) = \{10^{-1}, 10^{-2}, \mathbf{10^{-3}}, 10^{-4}\}$ $T_R = \{60, \mathbf{120}, 180\}$s	$T_S = 720$s $r_N = 3$m $T_N = 180$s
Maximum scouting time (Section A.3.2)	$T(S) = \{480, 600, 720, 840, 960,$ $\mathbf{1080}, 1200, 1320\}$s	$p(S) = 10^{-3}$ $T_R = 120$s $r_N = 3$m $T_N = 180$s
Neighbourhood search (Section A.3.3)	$r_N = \{1, 2, \mathbf{3}, 4\}$m $T_N = \{60, \mathbf{180}, 300\}$s	$p(S) = 10^{-3}$ $T_R = 120$s $T_S = 1080$s

Table A.3: Parameter groups for the following bee swarm parameters: scouting probability, $p(S)$, recruitment time, T_R, maximum scouting time, T_S, neighbourhood search radius, r_N and neighbourhood search time, T_N. Parameter groups were optimised in the order indicated by the table. During the optimisation, estimated suitable values were set for parameters from different groups ("Set parameters"). Parameter values that were selected as a result of optimisation are shown in bold.

A.3.1 Scouting versus recruitment

The scouting probability, $p(S)$, with which bee swarm observers left the base in order to scout, and the recruitment time, T_R, that determined how long a recruiters stayed in the base in order to interact with observers, were closely related. In swarms with a high value of $p(S)$, a very small number of observers could remain in the base, making recruitment more difficult. Similarly, a very short recruitment time resulted in a small number of interactions between observers and recruiters. On the other hand, a small

scouting probability or a long recruitment time resulted in frequent interactions between robots and a thus in a stronger recruitment.

In Scatter25 scenarios, where worksites were numerous, recruitment was not necessary as it often led to redundancy. In both the consumption and the collection tasks, using values of $(p(S), T_R) = (10^{-1}, 60\text{s})$ thus resulted in the best swarm performance, especially when worksites were close to the base (Figures A.3c and A.4c). On the other hand, recruitment was useful in the Heap1 scenarios, where only a single worksite was placed in the environment, since it allowed robots to take advantage of information that other members of the swarm had found. In the collection task, using a small scouting probability $(p(S) = 10^{-4})$ resulted in the largest amount of resource being collected

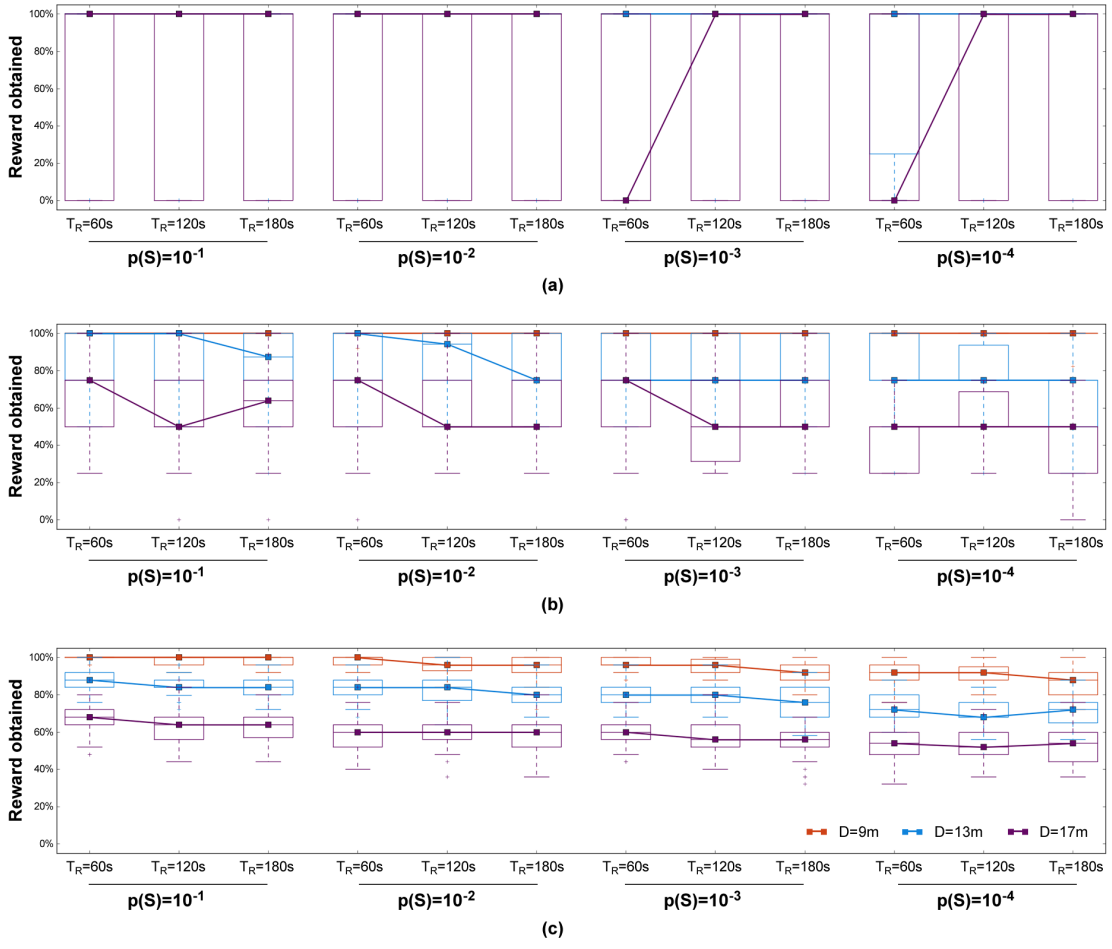


Figure A.3: The percentage of available reward obtained by 25-robot bee swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static consumption task, using various values for the scouting probability, $p(S)$, and the recruitment time, T_R, parameters. There was no clear pattern in the effect of the parameter values on the swarm performance in the Heap1 and Heap4 scenarios, where the performance was very variable due to the short simulation duration. In Scatter25, swarms with $p(S) = 10^{-1}$ and $T_R = 60\text{s}$ achieved the best performance.

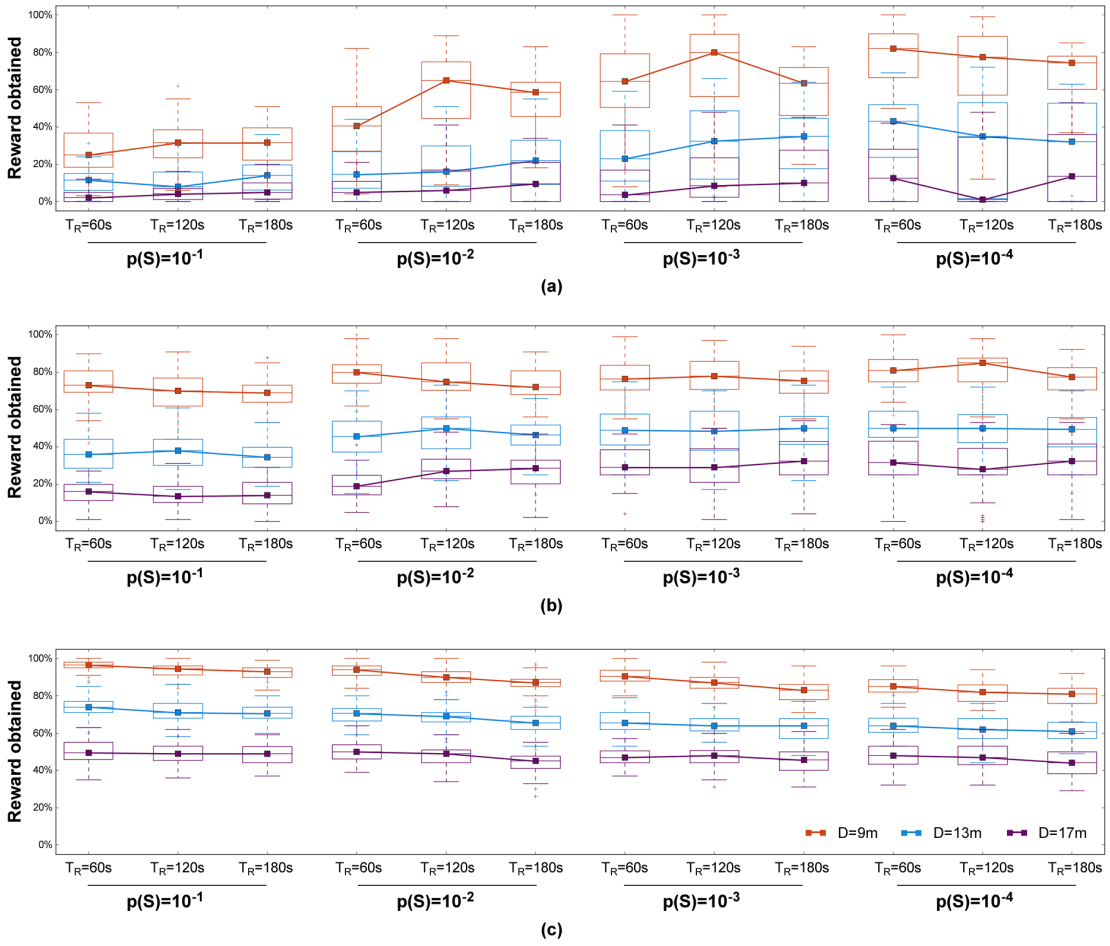


Figure A.4: The percentage of available reward obtained by 25-robot bee swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static collection task, using various values for the scouting probability, $p(S)$, and the recruitment time, T_R, parameters. In Heap1, swarms with $p(S) = 10^{-4}$ and $T_R = 60$s achieved the best performance. In Heap4, swarms with $p(S) \leq 10^{-2}$ had a similarly good performance. In Scatter25, the best performance was achieved with $p(S) = 10^{-1}$ and $T_R = 60$s.

(Figure A.4a). Since a lot of observers were usually present in the base, recruiters did not need to spend a long time recruiting and the best median performance was achieved when $T_R = 60$s, although the performance variance was usually fairly high. The one-hour consumption task in Heap1 was either fully completed or not completed at all, regardless of what parameter values were used (Figure A.3a).

Values of $(p(S), T_R) = (10^{-3}, 120\text{s})$ were selected as optimal for the bee swarms. Using these values led to a relatively high performance in the Heap1 and Heap4 scenarios during the collection tasks, while it did not significantly deteriorate the performance of bee swarms in the Scatter25 scenarios.

A.3.2 Maximum scouting time

Having established suitable values for scouting probability and recruitment time, the maximum scouting time, T_S was optimised in order to determine how long bee scouts should search the environment before returning to the base and becoming observers.

Setting a suitable value of T_S was more relevant when worksites were further away from the base, i.e., when it was less probable that a scout would find a worksite shortly after leaving the base. This was especially evident in Scatter25 scenarios, where a lot of worksites needed to be found. While the value of T_S had a very small impact on swarm performance when $D = 9$m, using the largest value of T_S from the explored value set was advantageous when $D = 17$m (Figures A.5c and A.6c). Similarly, higher values of T_S led to a better performance in Heap4 scenarios when worksite distance was large (Figures A.5b and A.6b).

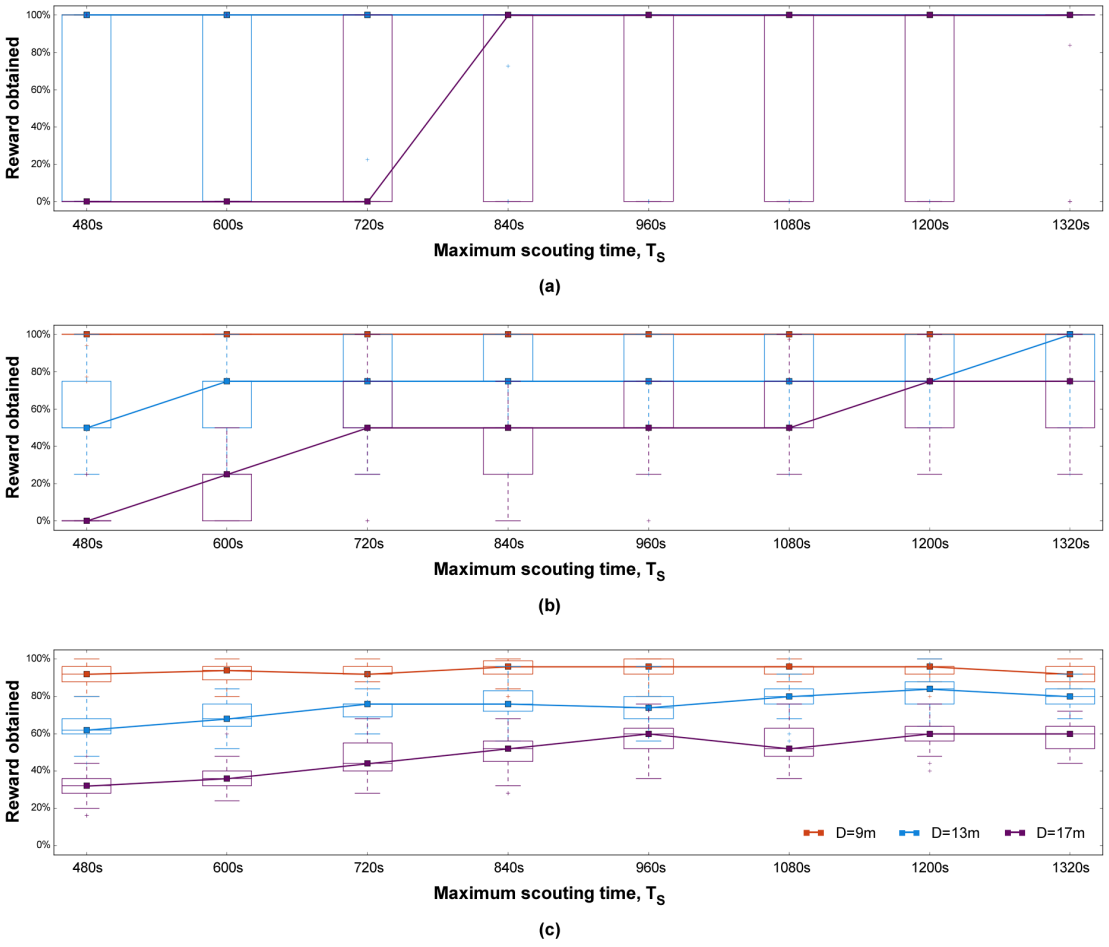


Figure A.5: The percentage of available reward obtained by 25-robot bee swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static consumption task, using various values for the scouting time, T_S, parameter. The parameter had the strongest effect on swarm performance when $D = 17$m. Higher values of T_S usually led to a better performance.

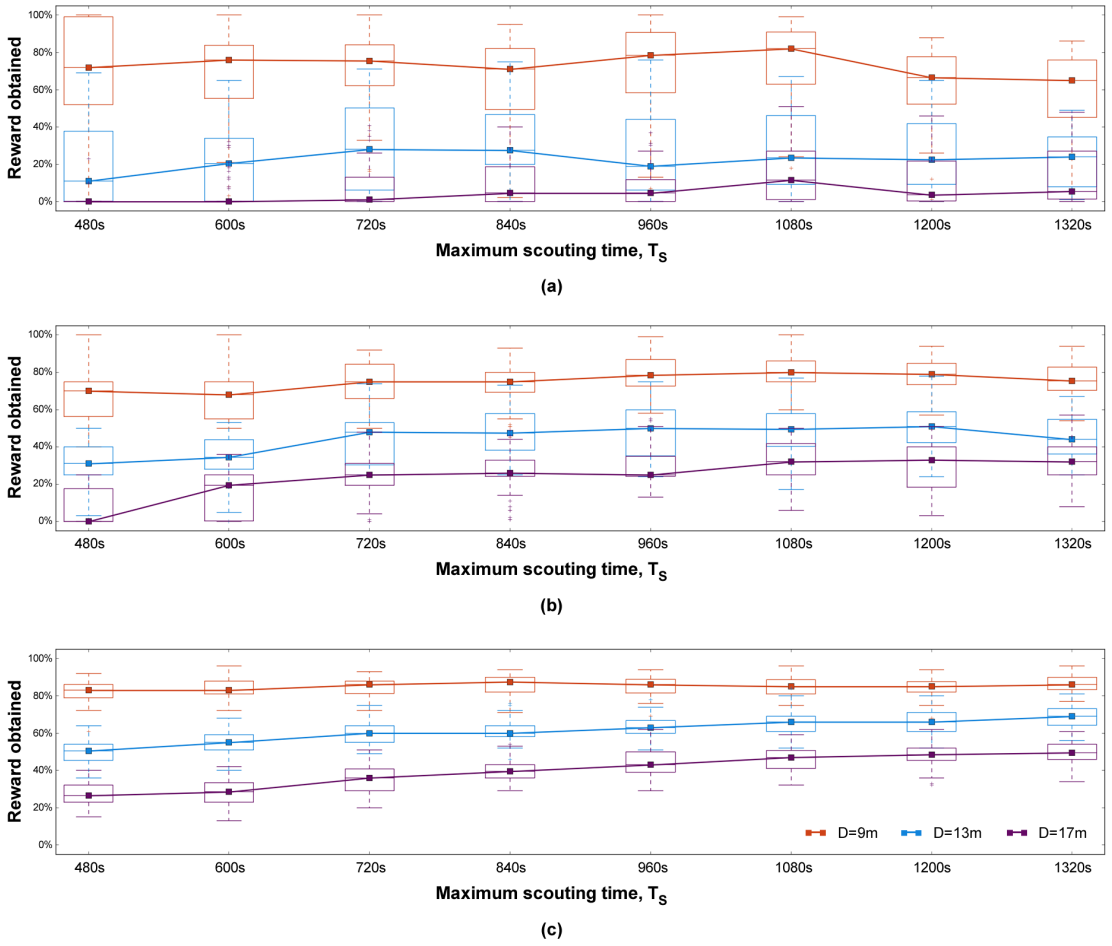


Figure A.6: The percentage of available reward obtained by 25-robot bee swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static collection task, using various values for the scouting time, T_S, parameter. Swarms with $T_S = 1080$s achieved the best performance in the Heap1 scenario. The performance usually increased or remained similar with an increasing T_S in the Heap4 and Scatter25 scenarios.

It was important to balance the amount of scouting and recruitment in Heap1 scenarios, where a worksite was difficult to find. Provided that there was enough observers in the base, only a single worksite discovery, followed by recruitment, was necessary in order to extract reward from the environment, especially in the collection task, where robots periodically returned to the base in order to unload resource. Using a very short maximum scouting time ($T_S \leq 600$s) prevented the robots from discovering the worksite and thus caused a poor performance (Figure A.6a). On the other hand, if the robots spent too much time as scouts ($T_S \geq 1200$s), there was not enough observers in the base to take the advantage of social information.

Based on these results, the value of $T_S = 1080$s was selected as optimal for bee swarms. It was one of the most suitable values in the Heap1 collection task, while it led to a

similar performance than that achieved with other high values of T_S in the Heap4 and Scatter25 scenarios during both swarm tasks.

A.3.3 Neighbourhood search

Since bee swarm robots were recruited in the base, odometry errors could accumulate during both swarm tasks. The r_N and T_N parameters were thus optimised for both the collection and the consumption tasks.

During the consumption task, bee swarm robots only recruited once, after a scout discovered a worksite, and working robots never left their worksites until they were depleted. Therefore, the parameter values did not have a significant effect on the swarm performance in any of the explored scenarios (Figure A.7). Similarly, there was no clear

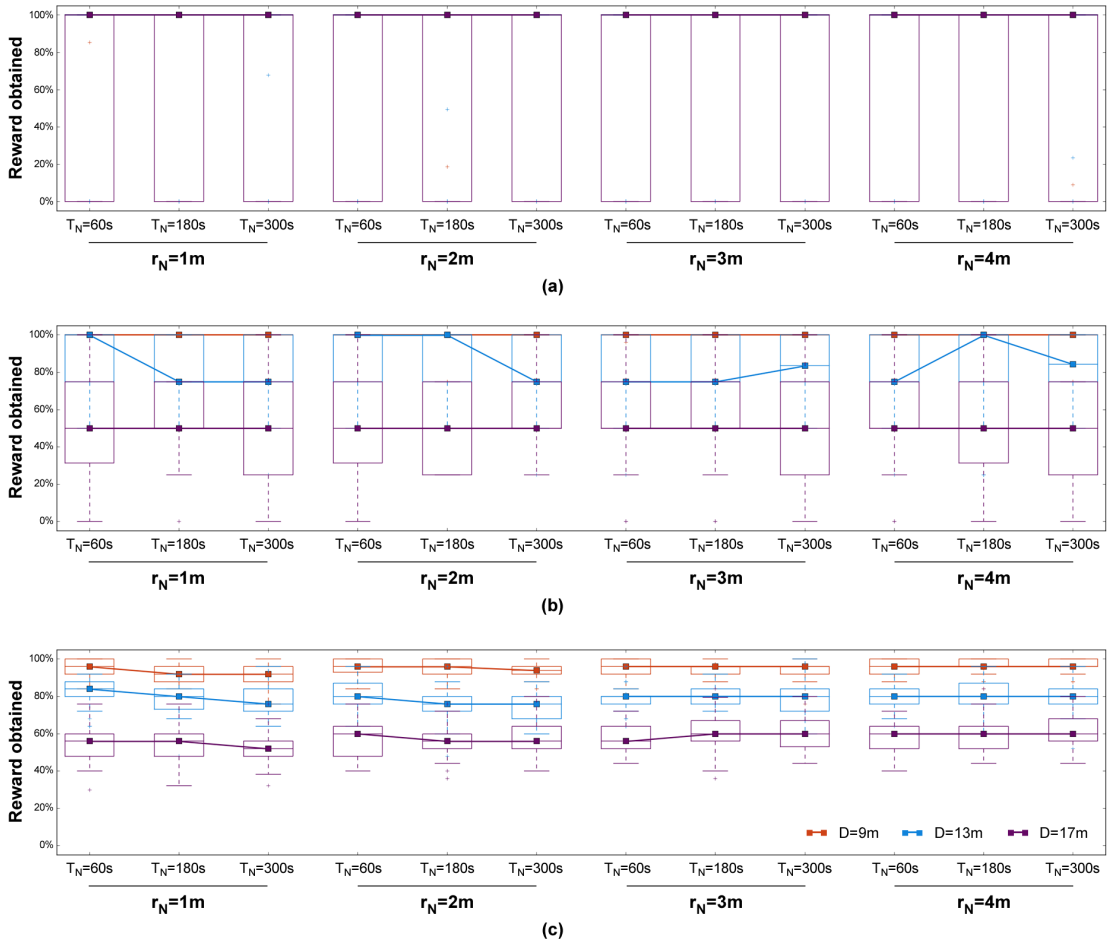


Figure A.7: The percentage of available reward obtained by 25-robot bee swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static consumption task, using various values for the neighbourhood search radius, r_N, and the neighbourhood search time, T_S, parameters. There was no clear pattern in the effect of the parameter values on the swarm performance.

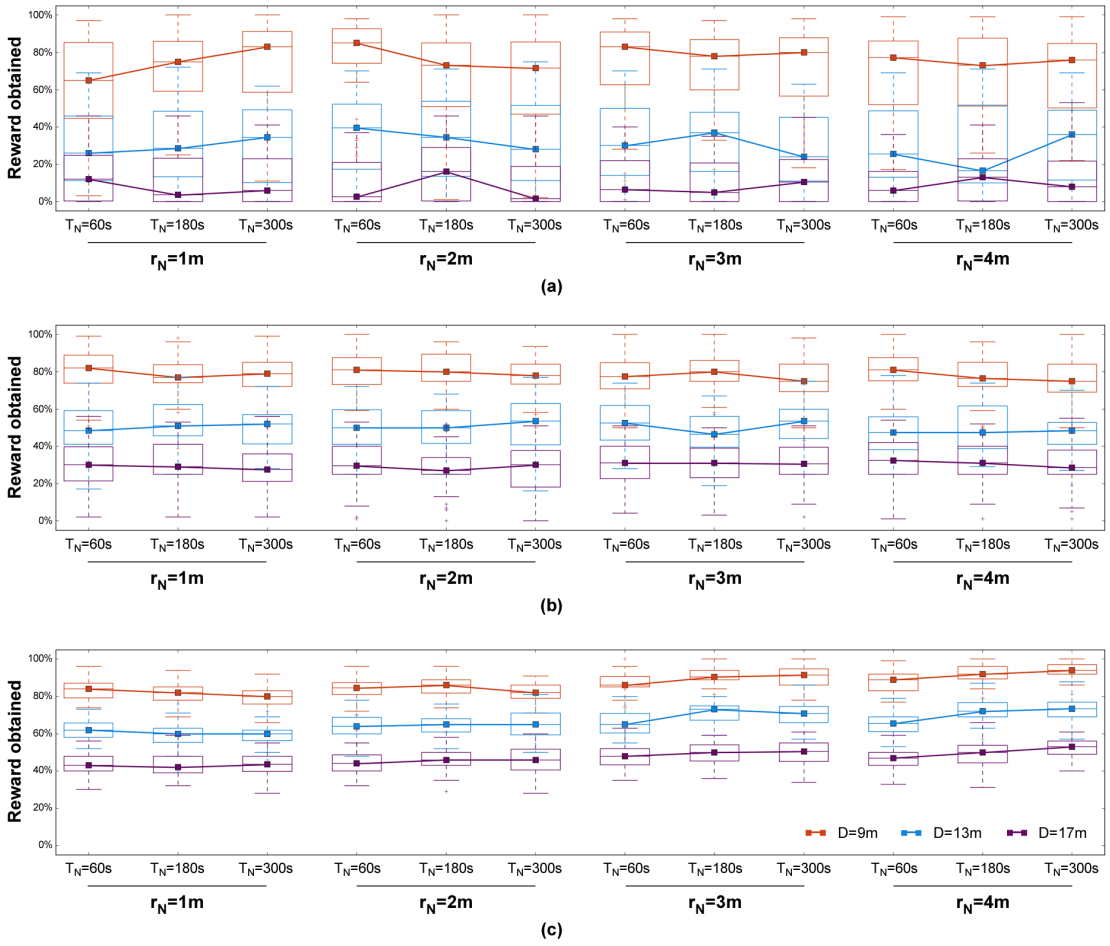


Figure A.8: The percentage of available reward obtained by 25-robot bee swarms in the (a) Heap1, (b) Heap4, (c) Scatter25 scenarios during the static collection task, using various values for the neighbourhood search radius, r_N, and the neighbourhood search time, T_S, parameters. There was no clear pattern in the effect of the parameter values on the swarm performance in the Heap1 and Heap4 scenarios. In Scatter25, swarms with $r_N = 4$m and $T_N = 300$s usually achieved the best performance.

pattern in the effect of these parameter values during the collection task in the Heap1 and Heap4 scenarios (Figure A.8a,b), where worksites had relatively large volumes and were therefore rarely depleted during a simulation run. On the other hand, during the collection task in the Scatter25 scenario, recruitment occurred often and worksites were depleted faster, since they had small volumes. In these scenarios, more extensive neighbourhood search ($(r_N, T_N) = (4\text{m}, 300\text{s})$) was advantageous (Figure A.8c).

In order to keep the neighbourhood search behaviour consistent with that of the other swarms, while preventing severe performance degradation in the Scatter25 collection task, values of $(r_N, T_N) = (3\text{m}, 180\text{s})$ were selected.

Appendix B

Supporting graphs for analysis of the static consumption task

B.1 Task completion time

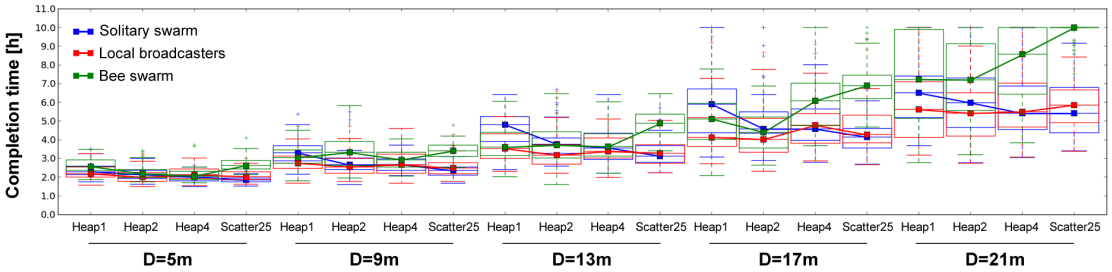


Figure B.1: The static consumption task completion time of 10-robot swarms. The swarms performed similarly well when worksite distance was small. Bee swarms took longer to complete the task when worksite distance is large.

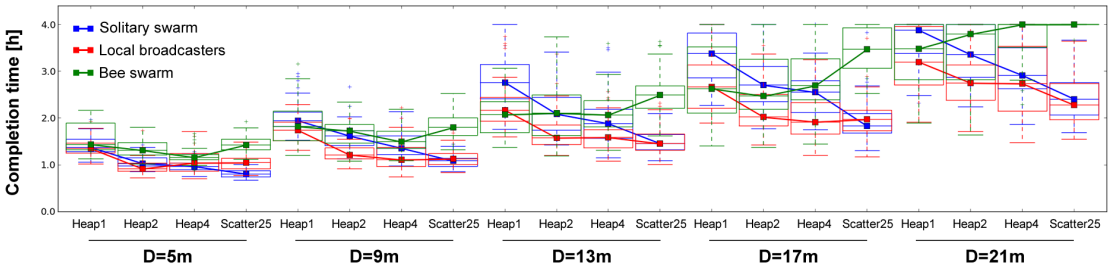


Figure B.2: The static consumption task completion time of 25-robot swarms. Local broadcaster and solitary swarms outperformed the bee swarms in most environments. Local broadcasters were the best performing control strategy in most Heap environments. Solitary swarms outperformed the other swarms in Scatter25 scenario when $D = 5$m.

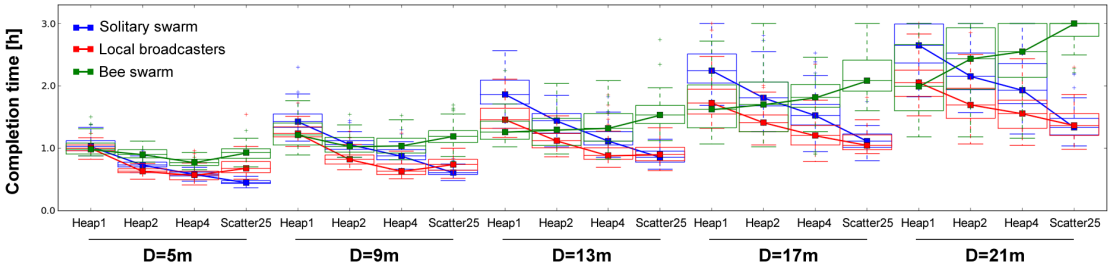


Figure B.3: The static consumption task completion time of 50-robot swarms. Local broadcaster and solitary swarms outperformed the bee swarms in most environments. Local broadcasters were the best performing control strategy in most Heap environments. Solitary swarms outperformed the other swarms in Scatter25 scenario when $D \leq 9$m.

B.2 The first worksite discovery

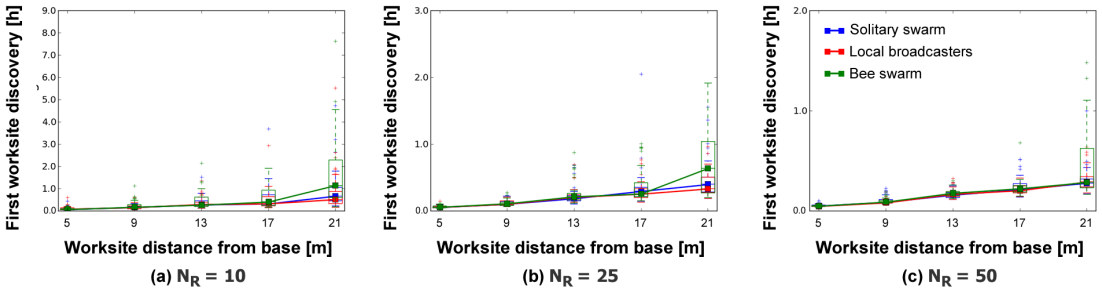


Figure B.4: Time of the first worksite discovery in the static consumption task, Heap2 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other swarms.

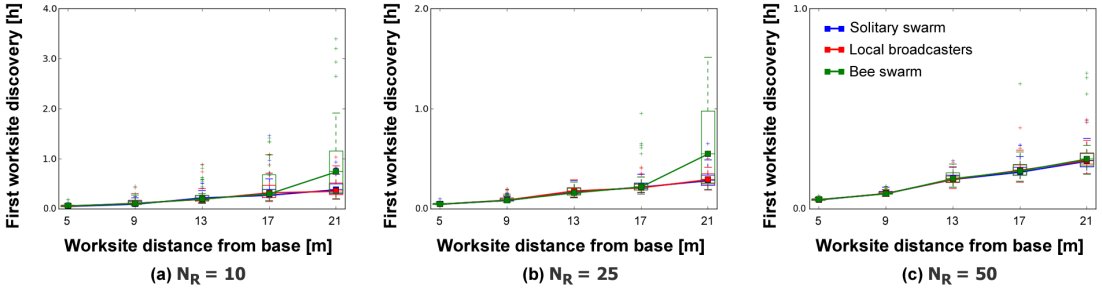


Figure B.5: Time of the first worksite discovery in the static consumption task, Heap4 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other swarms.

B.3 Information gain rate

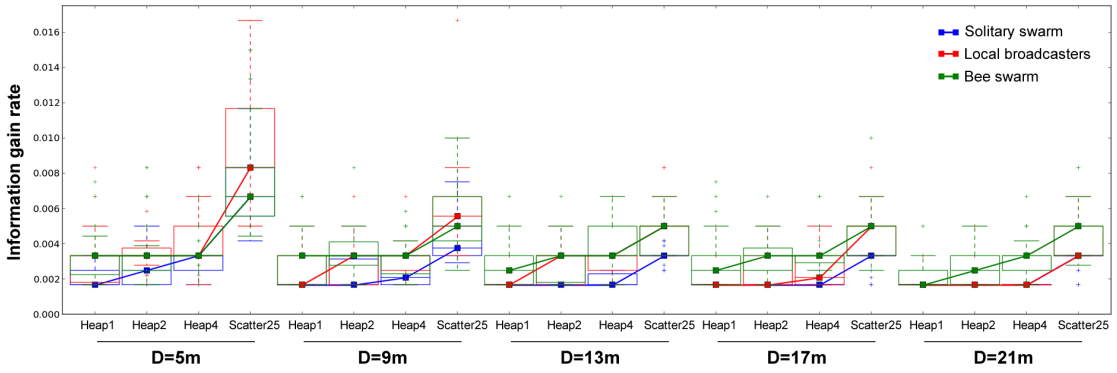


Figure B.6: Information gain rate, i, of 10-robot swarms in the static consumption task. Bee swarms enjoyed the highest i in most environments, apart from the Scatter25 scenarios when $D \leq 9$m.

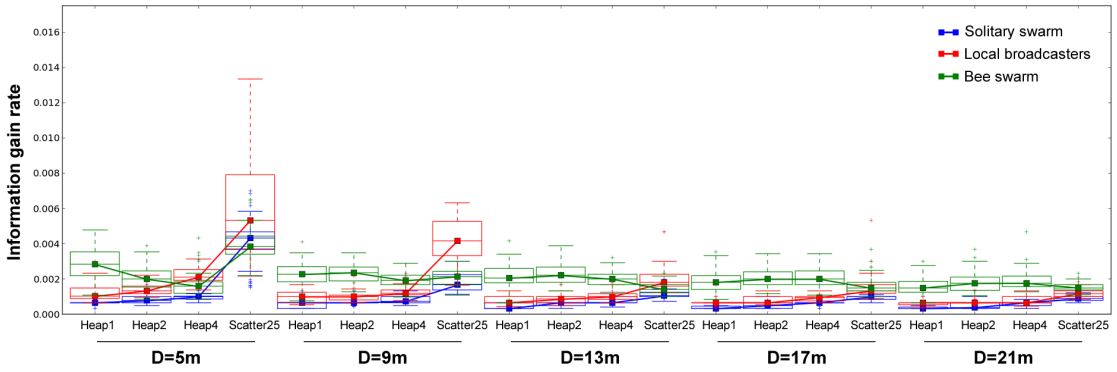


Figure B.7: Information gain rate, i, of 50-robot swarms in the static consumption task. Bee swarms enjoyed the highest i in most environments, apart from the Heap4 scenario when $D = 5$m and the Scatter25 scenarios when $D \leq 13$m.

B.4 Misplacement cost coefficient

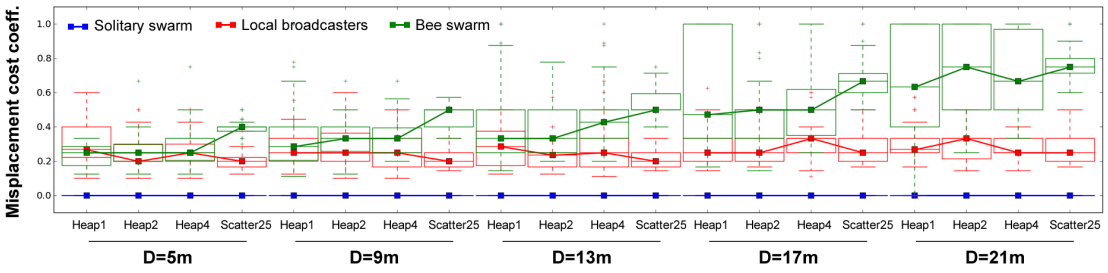


Figure B.8: Misplacement cost coefficient, m, of 10-robot swarms in the static consumption task. Bee swarms usually had the highest m, apart from environments where the number of worksites and worksite distance were small. The m of solitary swarms was always 0.

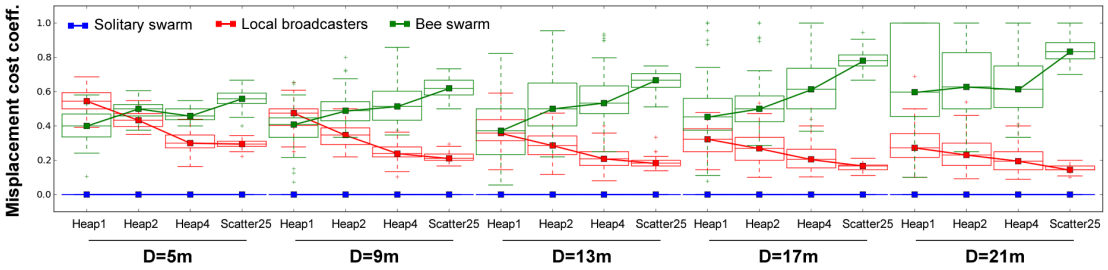


Figure B.9: Misplacement cost coefficient, m, of 50-robot swarms in the static consumption task. Bee swarms usually had the highest m, apart from environments where the number of worksites and worksite distance were small. The m of solitary swarms was always 0.

B.5 Opportunity cost

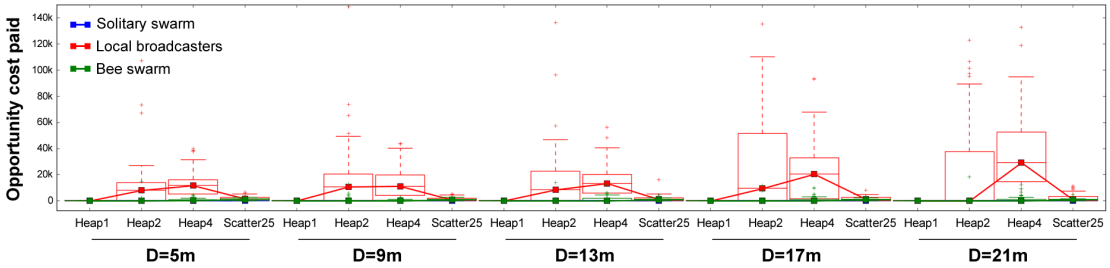


Figure B.10: Opportunity cost, C_O, of 10-robot swarms in the static consumption task. Local broadcasters paid a larger C_O than the other swarms in Heap environments, while in Scatter25 environments, C_O of bee swarms and local broadcasters was similar. Solitary swarms did not pay C_O.

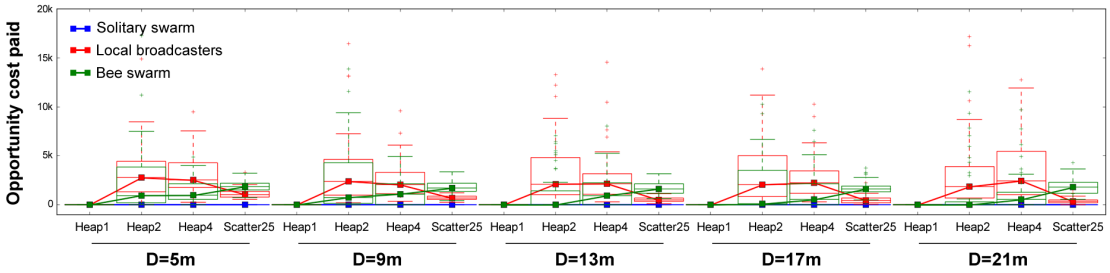


Figure B.11: Opportunity cost, C_O, of 50-robot swarms in the static consumption task. Local broadcasters paid a larger C_O than the other swarms in Heap environments, while in Scatter25 environments, C_O of bee swarms was the largest. Solitary swarms did not pay C_O.

Appendix C

Parameter sensitivity analysis for the information gain down-sampling interval, T_i

Calculation of the information gain rate, i, involves down-sampling the information gain time series into bins $T_i = 60$s long, identifying positive information gain regions based on the sampled down time series, and calculating the ratio between the total information gain in these regions and the length of the regions (see Section 4.2.3). In this Appendix, the effect of the value selected for T_i on the measured information gain rate of swarms is investigated.

Large values of T_i decrease the measured information gain rate, since the same total information gain can potentially get divided by a longer time period (Figures C.1 – C.6). Consequently, values of T_i, such as $T_i = 120$s and $T_i = 240$s, decrease the difference in i between controllers.

On the other hand, setting T_i to be very small ($T_i = 30$s) increases the median information gain rate, but also makes the distribution of measured information gain rates across multiple runs in the same scenario less normal, especially in environments where worksites are far away from each other (i.e., when D is large). This is because positive regions of a certain length, characteristic for a particular combination of scenario and worksite distance, appear more often than other regions, making it difficult to find a normal distribution across an experimental run.

Nevertheless, if the investigated control strategies are ordered by their information gain rates, the order does not change when different values of T_i are used. Using $T_i = 60$s makes differences in i of the swarms as clear as possible, while avoiding non-normal i distributions.

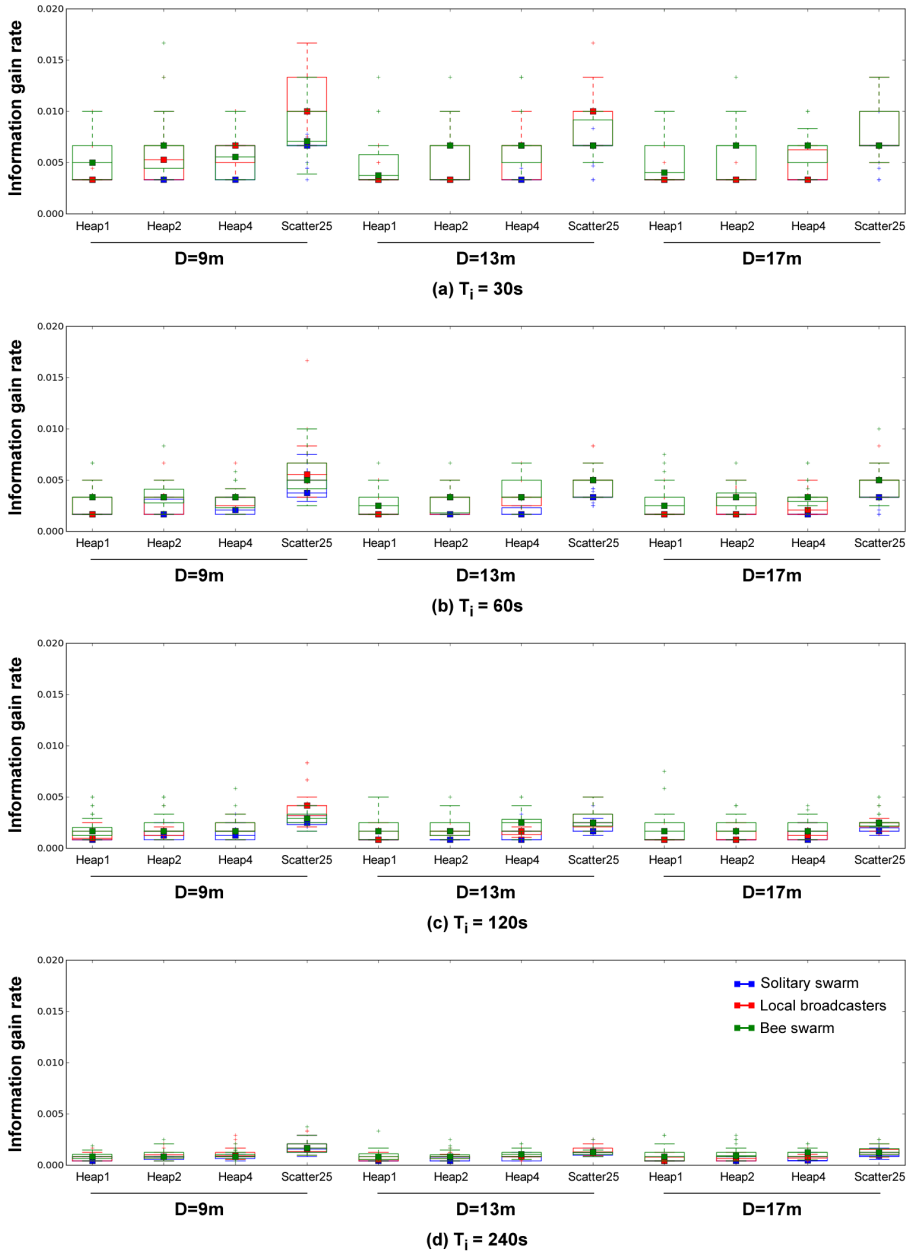


Figure C.1: Information gain rate of 10-robot swarms in the static consumption task, various environments, using (a) $T_i = 30$s, (b) $T_i = 60$s, (c) $T_i = 120$s, (d) $T_i = 240$s.

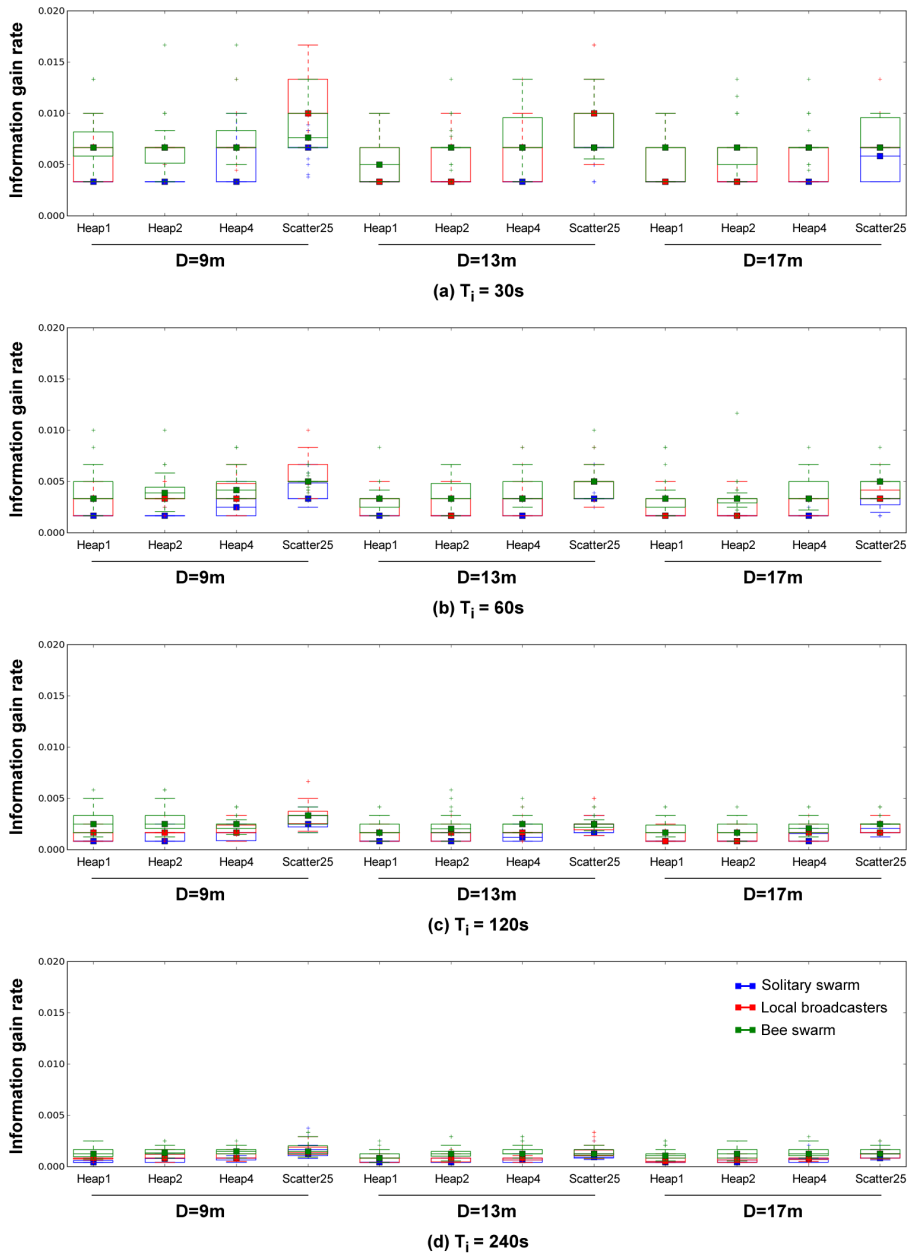


Figure C.2: Information gain rate of 10-robot swarms in the static collection task, various environments, using (a) $T_i = 30$s, (b) $T_i = 60$s, (c) $T_i = 120$s, (d) $T_i = 240$s.

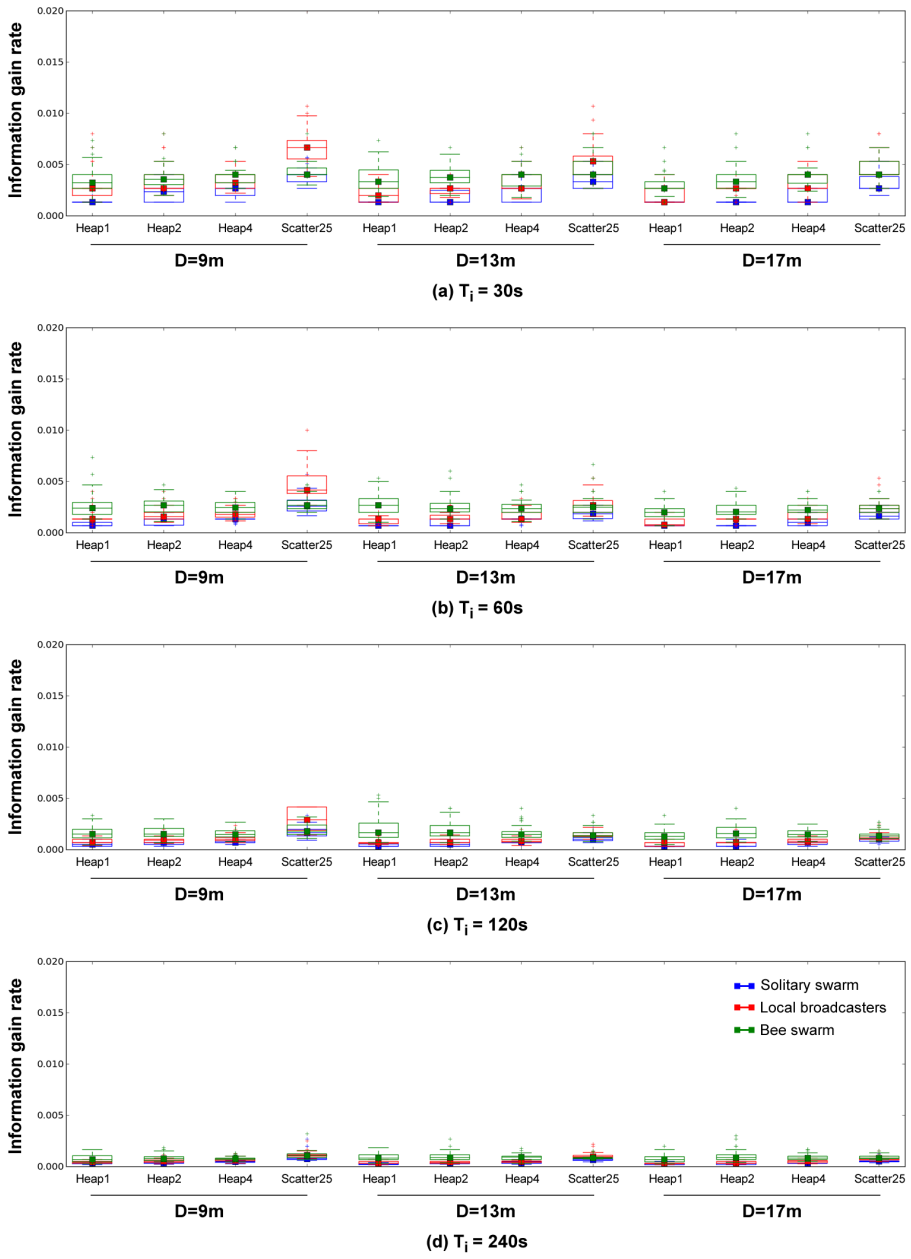


Figure C.3: Information gain rate of 25-robot swarms in the static consumption task, various environments, using (a) $T_i = 30$s, (b) $T_i = 60$s, (c) $T_i = 120$s, (d) $T_i = 240$s.

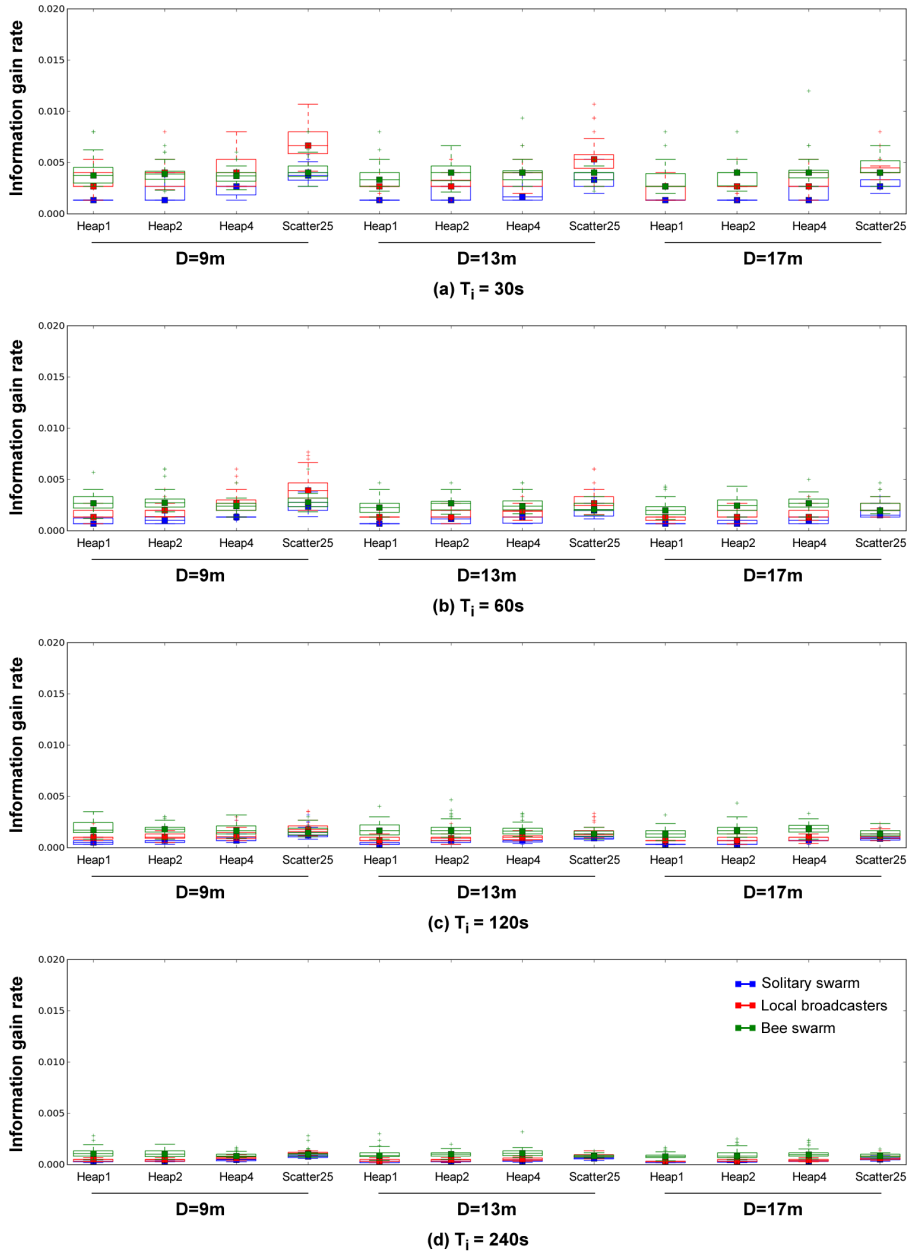


Figure C.4: Information gain rate of 25-robot swarms in the static collection task, various environments, using (a) $T_i = 30$s, (b) $T_i = 60$s, (c) $T_i = 120$s, (d) $T_i = 240$s.

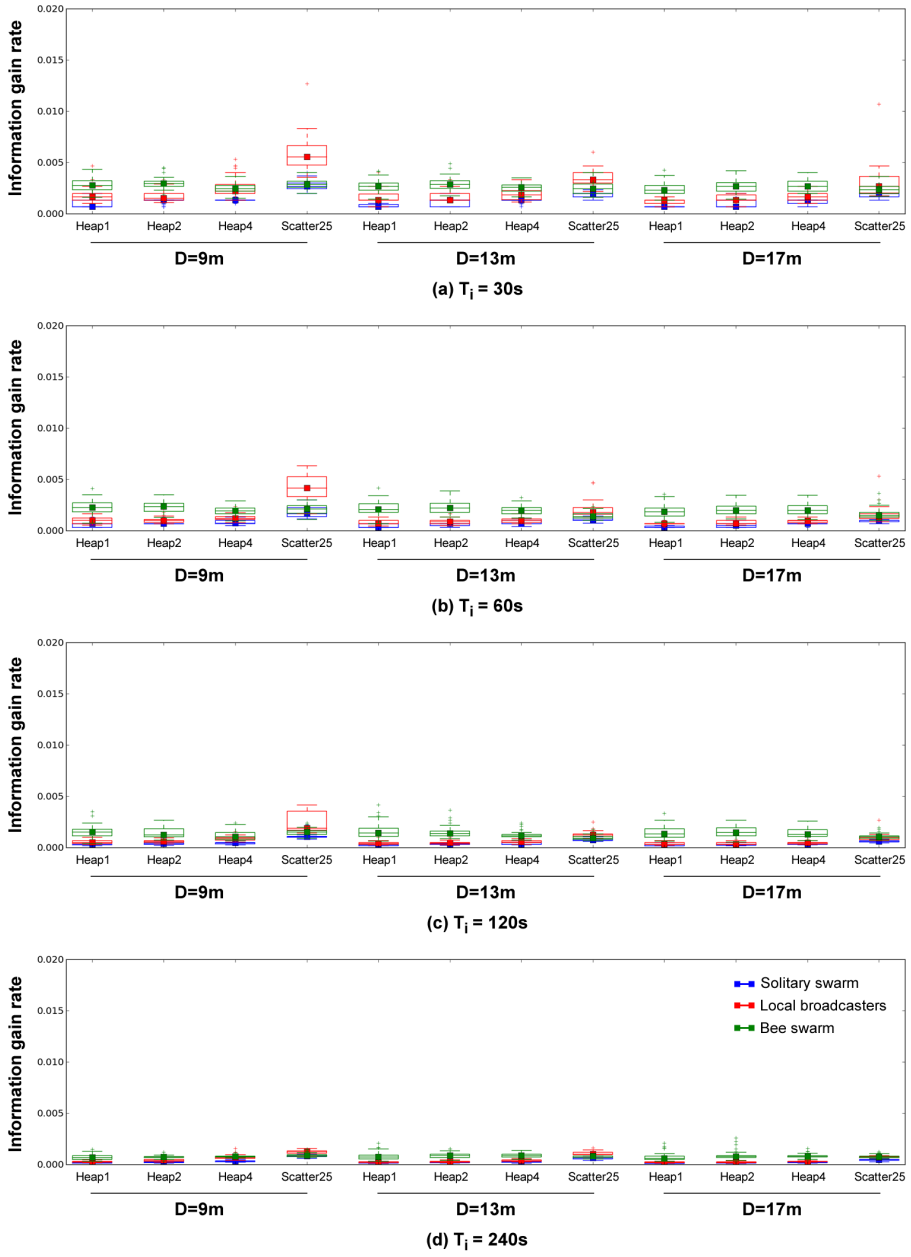


Figure C.5: Information gain rate of 50-robot swarms in the static consumption task, various environments, using (a) $T_i = 30$s, (b) $T_i = 60$s, (c) $T_i = 120$s, (d) $T_i = 240$s.

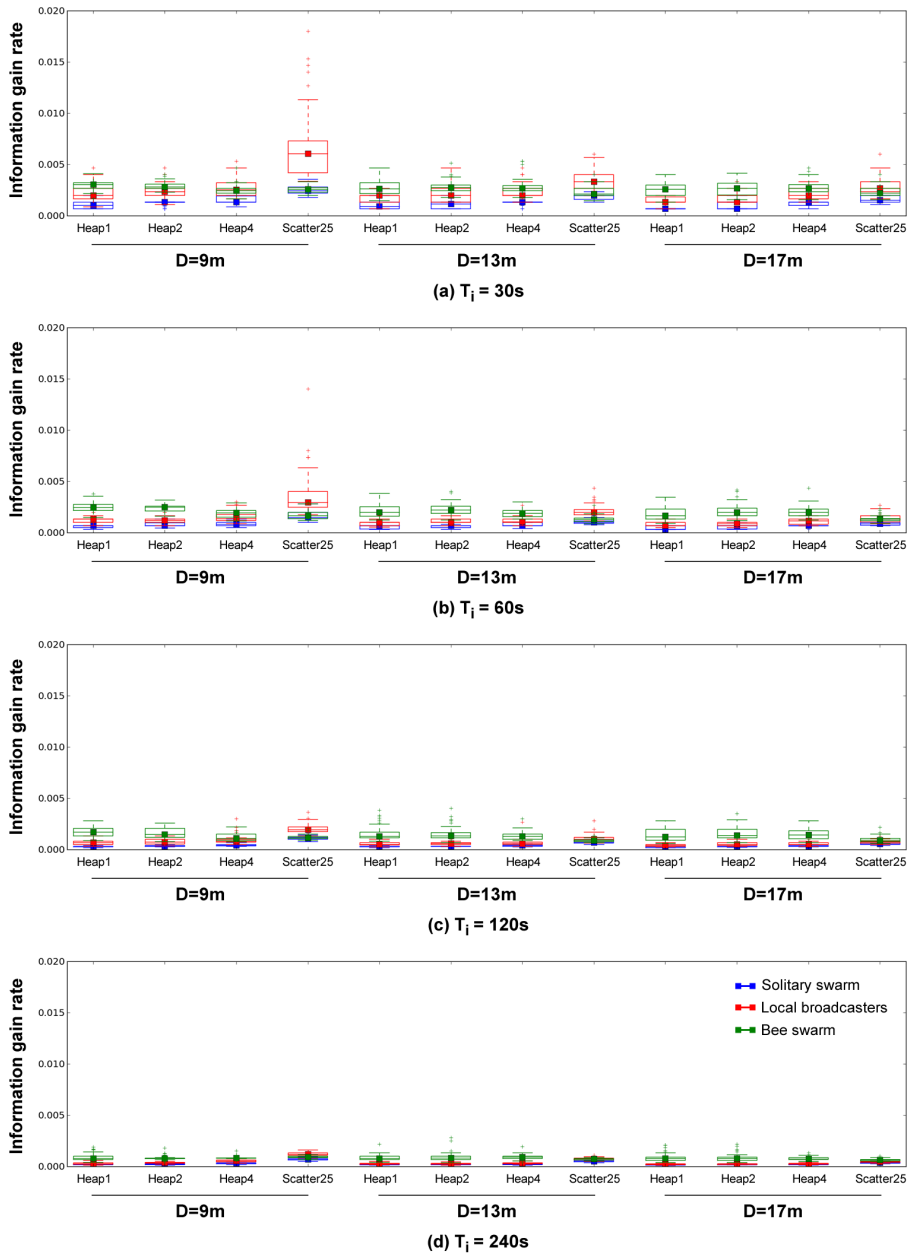


Figure C.6: Information gain rate of 50-robot swarms in the static collection task, various environments, using (a) $T_i = 30$s, (b) $T_i = 60$s, (c) $T_i = 120$s, (d) $T_i = 240$s.

Appendix D

Supporting graphs for the analysis of the static collection task

D.1 The first worksite discovery

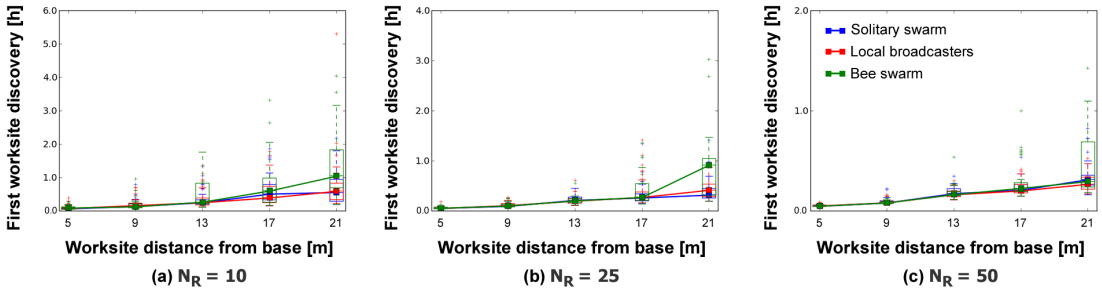


Figure D.1: Time of the first worksite discovery in the static collection task, Heap2 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, $D = 21$m, was larger in bee swarms, compared to the other swarms when $N_R \leq 25$.

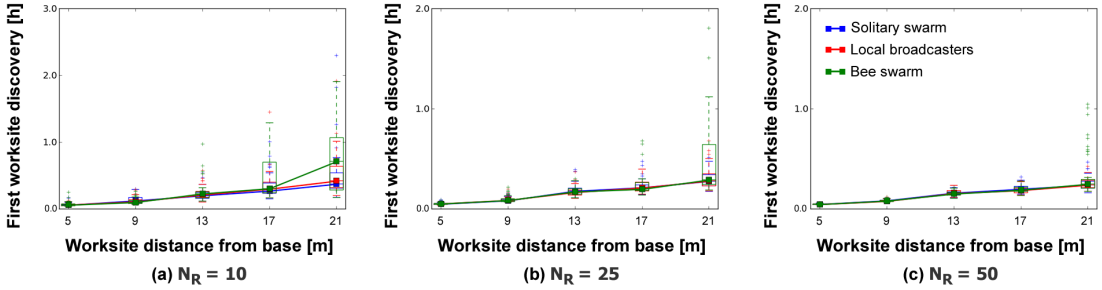


Figure D.2: Time of the first worksite discovery in the static collection task, Heap4 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, $D = 21$m, was larger in bee swarms, compared to the other swarms when $N_R = 10$.

D.2 Information gain rate

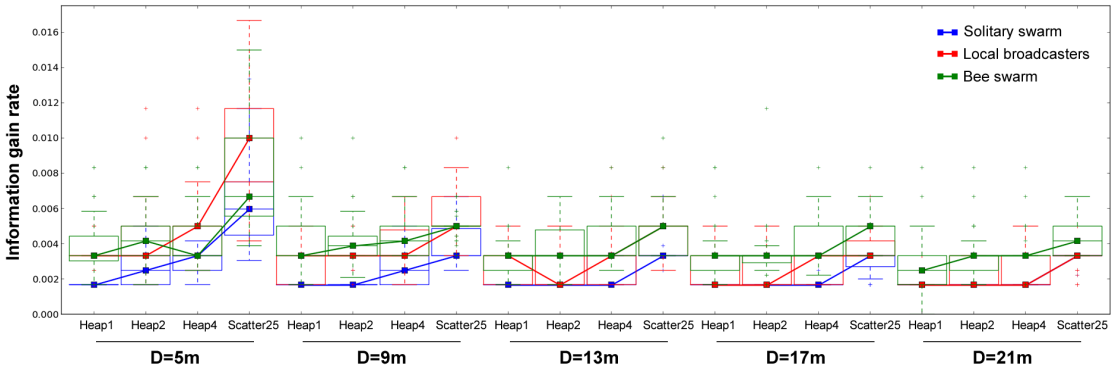


Figure D.3: Information gain rate, i, of 10-robot swarms in the static collection task. Bee swarms and local broadcaster swarms enjoyed a higher i than solitary swarms in most environments.

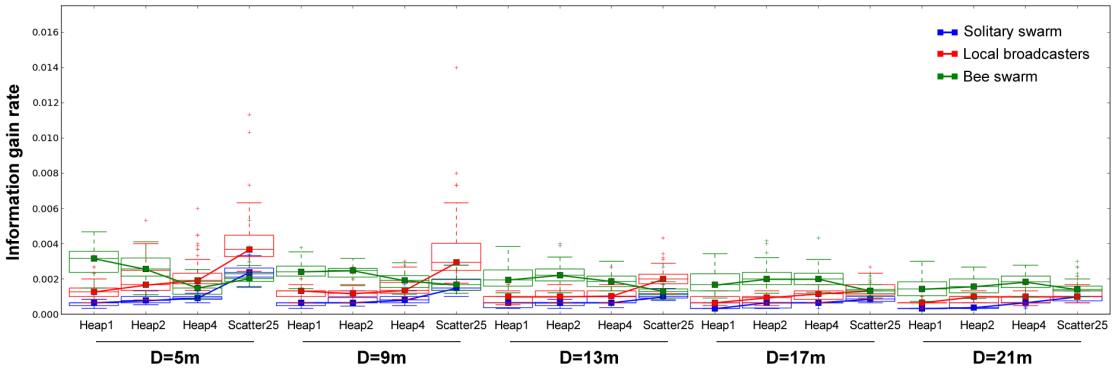


Figure D.4: Information gain rate, i, of 50-robot swarms in the static collection task. Bee swarm enjoyed the highest i in most environments, apart from the Scatter25 scenarios when worksite distance $D \leq 17$m and the Heap4 scenarios when $D \leq 5$m.

D.3 Misplacement cost coefficient

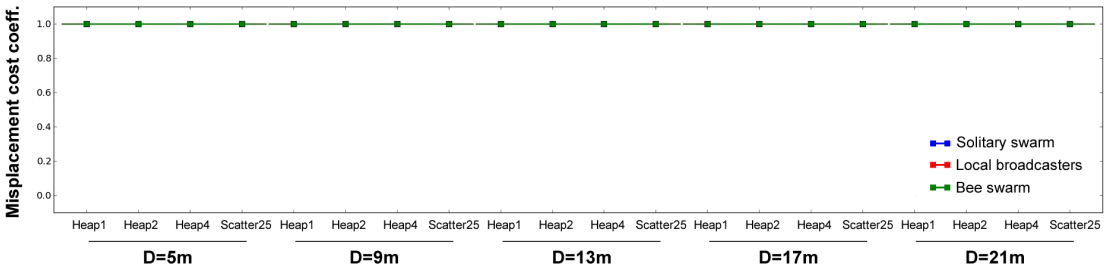


Figure D.5: Misplacement cost coefficient, m, of 10-robot swarms in the static collection task. Because robots needed to travel to the base to obtain reward, m of all swarms was equal to 1.

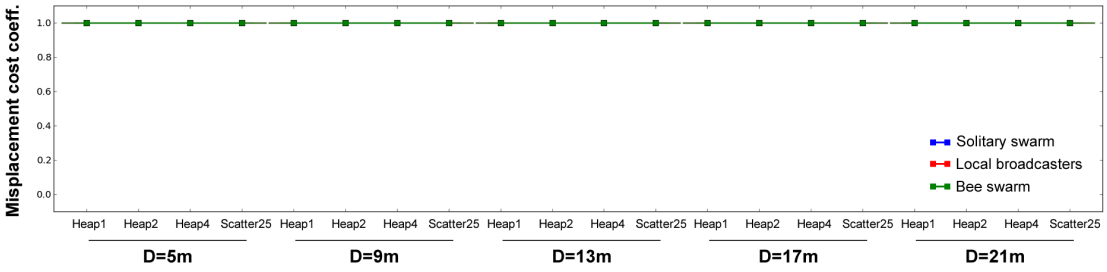


Figure D.6: Misplacement cost coefficient, m, of 50-robot swarms in the static collection task. Because robots needed to travel to the base to obtain reward, m of all swarms was equal to 1.

D.4 Opportunity cost

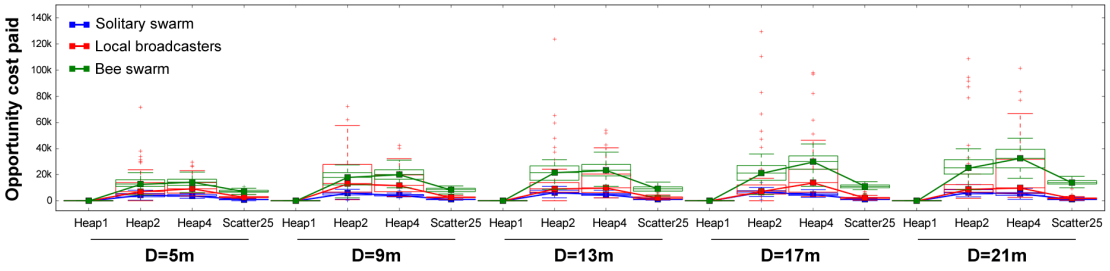


Figure D.7: Opportunity cost, C_O, of 10-robot swarms in the static collection task. Bee swarms paid the largest amount of C_O, followed by local broadcasters and solitary swarms. C_O of all swarms increased with worksite distance, D, given the same scenario type.

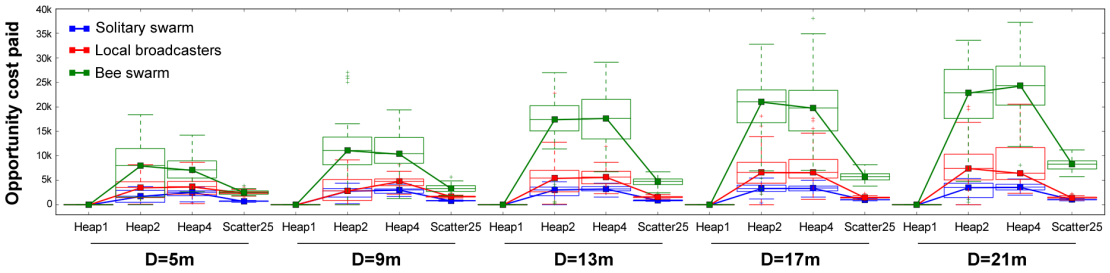


Figure D.8: Opportunity cost, C_O, of 50-robot swarms in the static collection task. Bee swarms paid the largest amount of C_O, followed by local broadcasters and solitary swarms. C_O of all swarms increased with worksite distance, D, given the same scenario type.

D.5 Task completion time

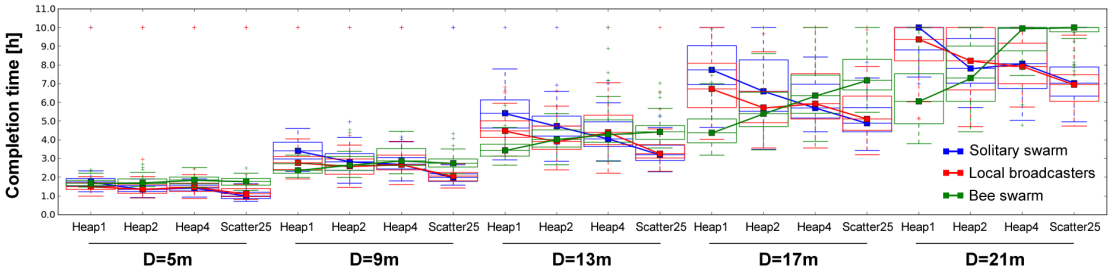


Figure D.9: Static collection task completion time of 10-robot swarms. Bee swarms enjoyed a performance advantage over the other swarms in Heap1 environments. Solitary and local broadcaster were more suitable in environments with a large number of worksites.

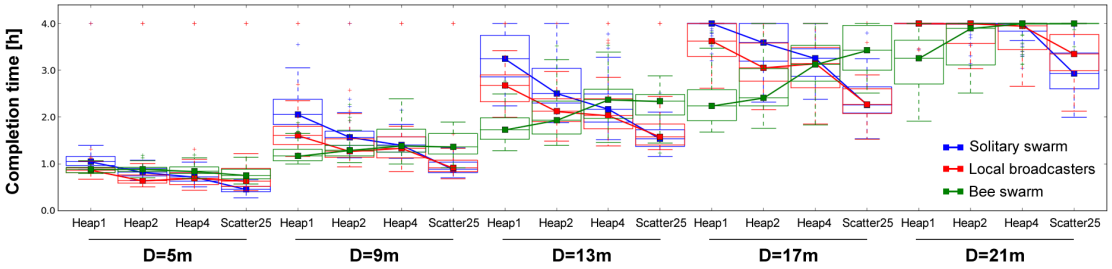


Figure D.10: The static collection task completion time of 25-robot swarms. Bee swarms enjoyed a performance advantage over the other swarms in Heap1 and some Heap2 environments. Solitary and local broadcaster were more suitable in environments with a large number of worksites.

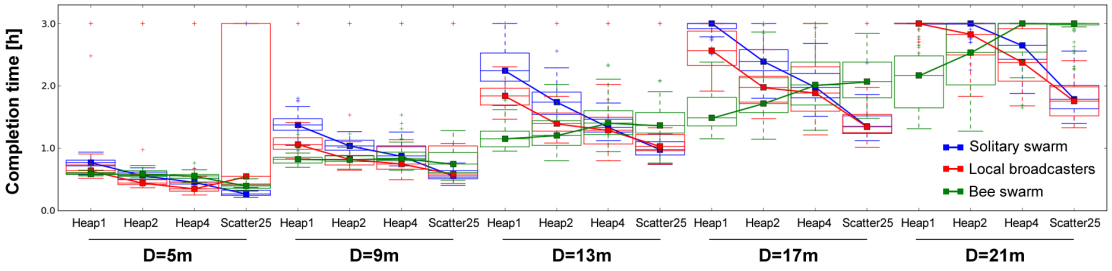


Figure D.11: The static collection task completion time of 50-robot swarms. Bee swarms enjoyed a performance advantage over the other swarms in Heap1 and some Heap2 environments. Solitary and local broadcaster were more suitable in environments with a large number of worksites.

Note the high variance in the performance of local broadcasters in Scatter25 environments when the swarm is large and worksites distance, D, is small (Figure D.11). The high variance is caused by congestion around worksites that results from the nature of the collection task. Local broadcasters enjoy the highest information gain rate compared to the other swarms in these environments, meaning that a lot of robots are usually recruited to the same worksite. Robots that are returning to the base to unload resource prevent other workers from returning to worksites. In some runs, a high amount of congestion around worksites prevents the robots from exploring the environment and leads to completion times significantly higher than that of the other swarms and that of local broadcasters in the consumption task.

Appendix E

Supporting graphs for the analysis of the dynamic consumption and collection tasks

E.1 The first worksite discovery

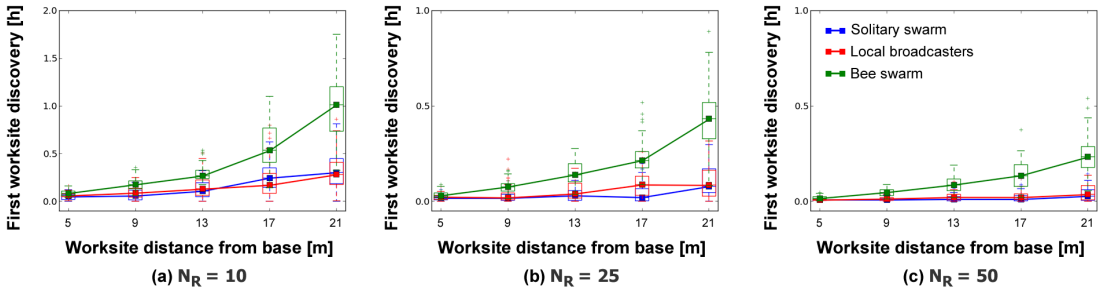


Figure E.1: Time of the first worksite discovery in the slow dynamic consumption task, Heap2 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other swarms.

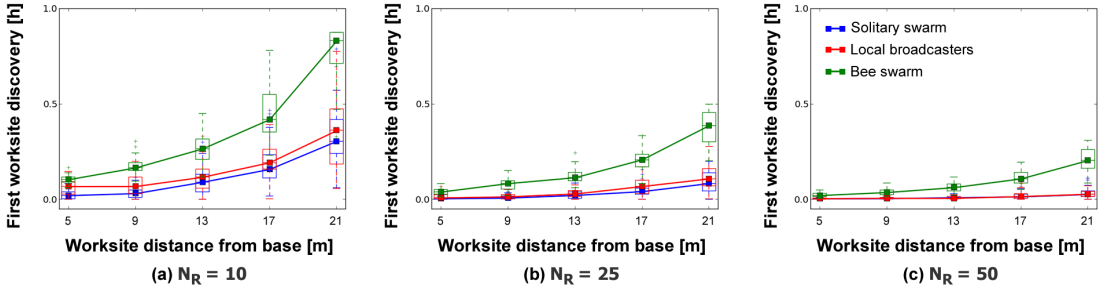


Figure E.2: Time of the first worksite discovery in the fast dynamic consumption task, Heap2 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms in most environments, compared to the other swarms.

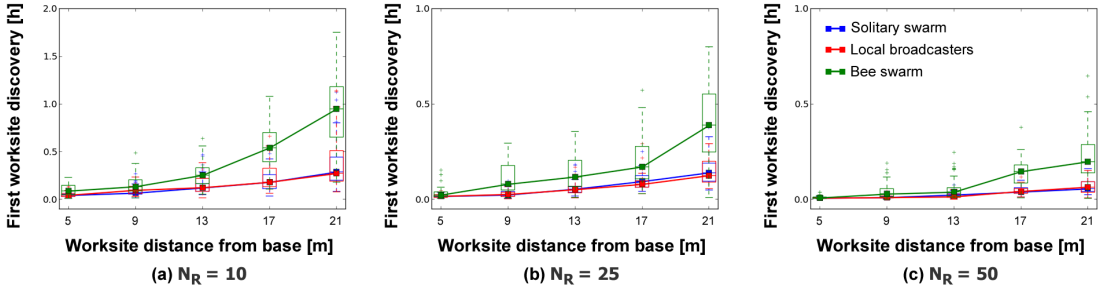


Figure E.3: Time of the first worksite discovery in the slow dynamic collection task, Heap2 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other swarms.

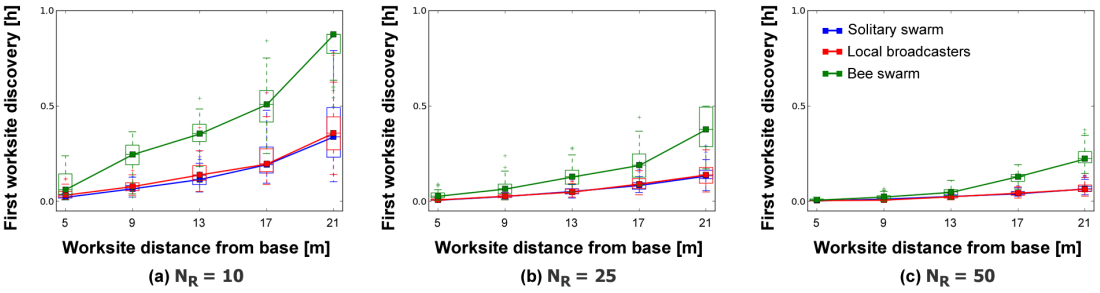


Figure E.4: Time of the first worksite discovery in the fast dynamic collection task, Heap2 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms in most environments, compared to the other swarms.

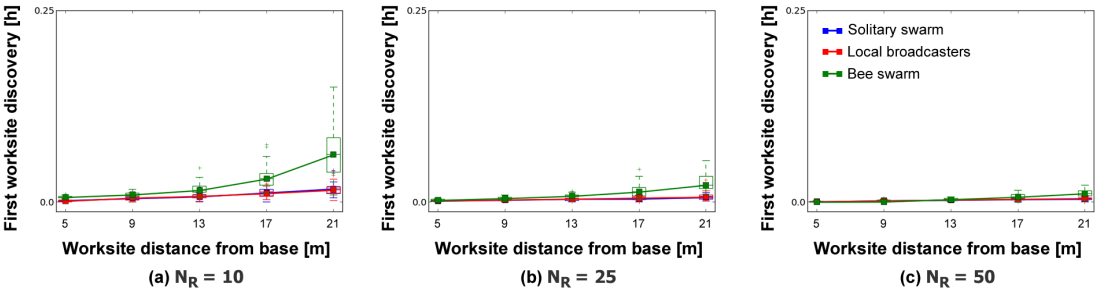


Figure E.5: Time of the first worksite discovery in the slow dynamic consumption task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other swarms, especially when $N_R = 10$.

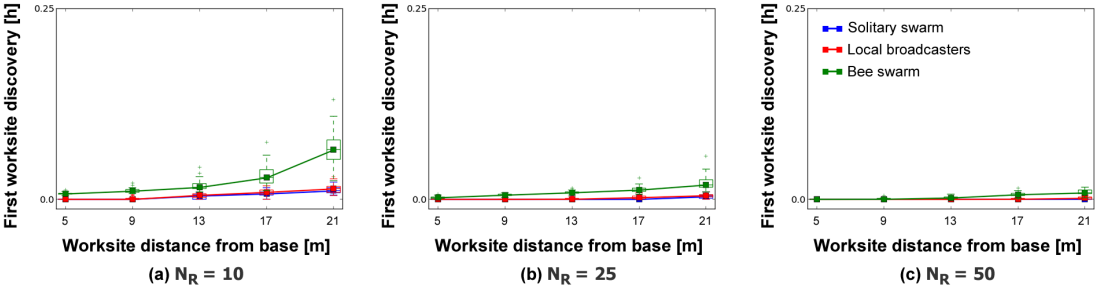


Figure E.6: Time of the first worksite discovery in the fast dynamic consumption task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms in most environments, compared to the other swarms, especially when $N_R = 10$.

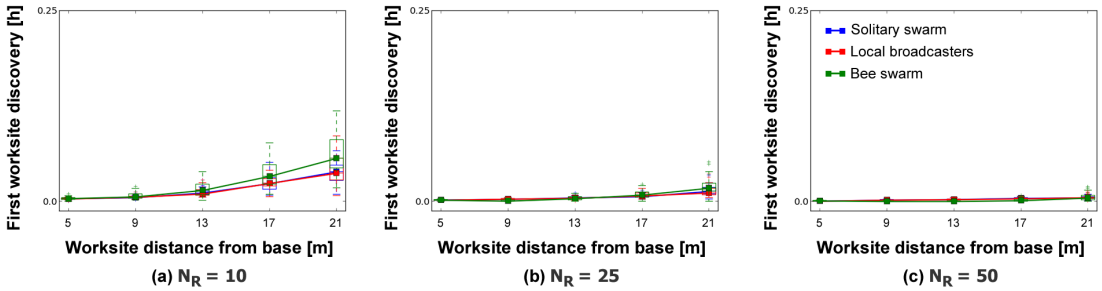


Figure E.7: Time of the first worksite discovery in the slow dynamic collection task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was larger in bee swarms, compared to the other swarms when $N_R = 10$.

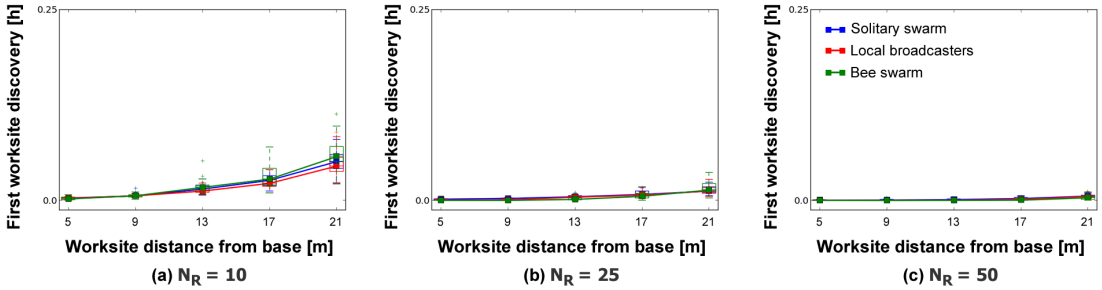


Figure E.8: Time of the first worksite discovery in the fast dynamic collection task, Scatter25 environments, using (a) 10, (b) 25 and (c) 50 robots. The negative effect of large worksite distance, D, was similar for all swarms.

E.2 Information gain rate

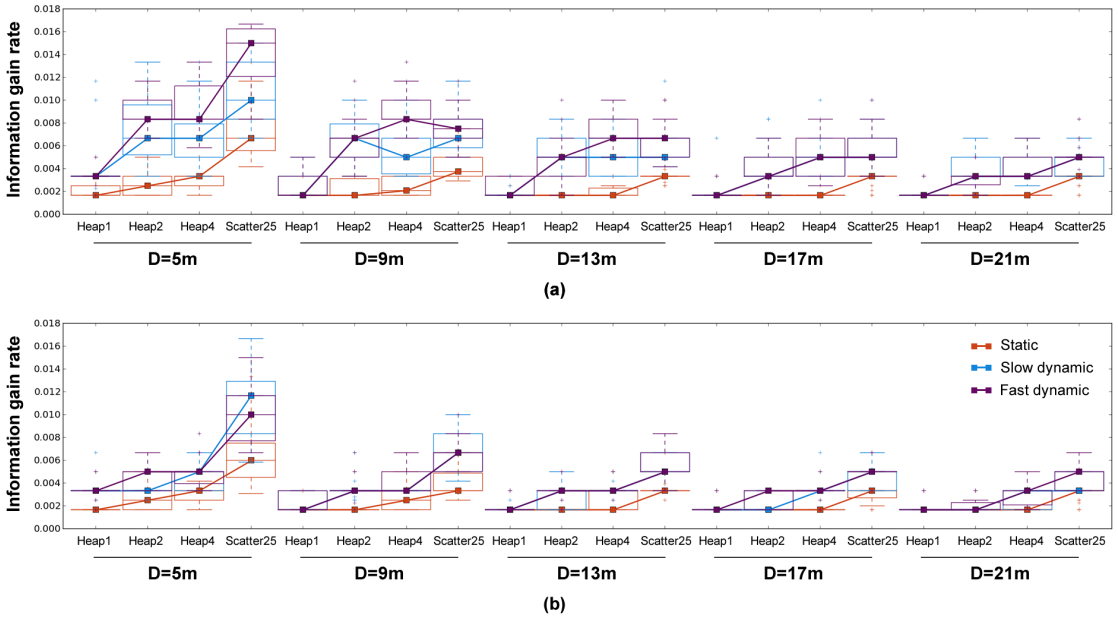


Figure E.9: Information gain rate, i, of 10-robot solitary swarms in the (a) consumption, (b) collection task. The information gain rate was higher in dynamic, compared to static environments, most notably in the consumption task or when worksite distance, D was small.

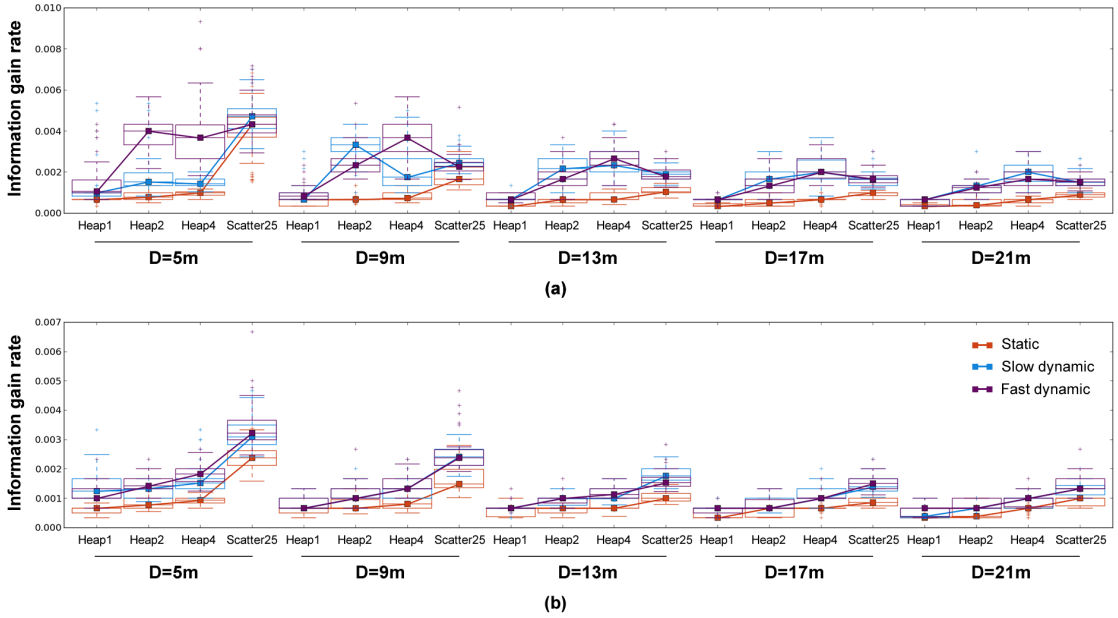


Figure E.10: Information gain rate, i, of 50-robot solitary swarms in the (a) consumption, (b) collection task. The information gain rate was higher in dynamic, compared to static environments, most notably in the fast dynamic consumption task or when worksite distance, D was small.

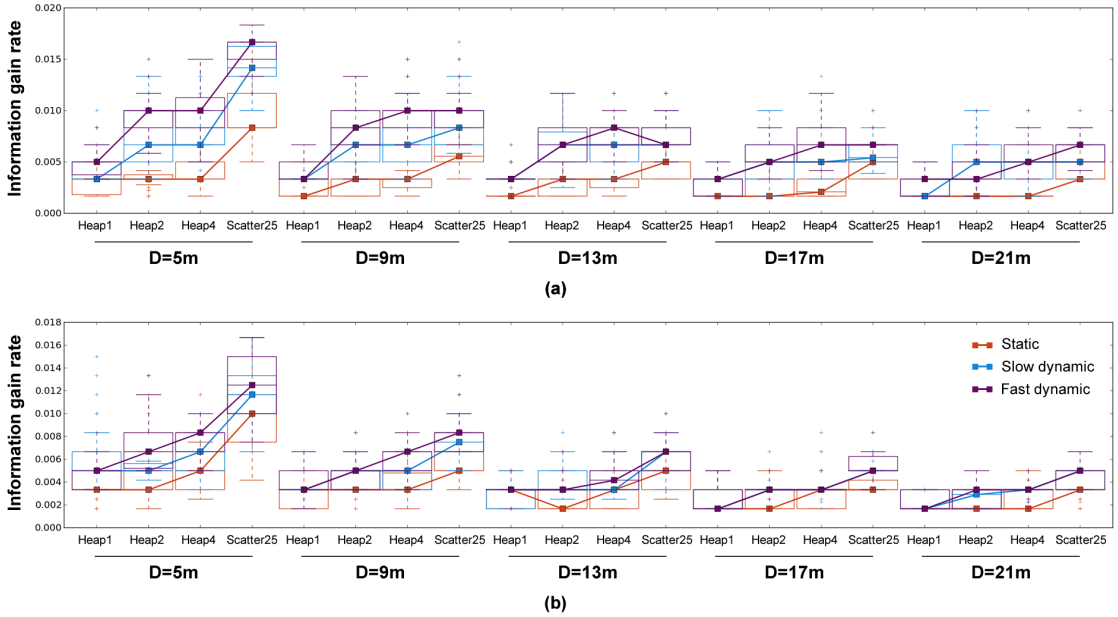


Figure E.11: Information gain rate, i, of 10-robot local broadcaster swarms in the (a) consumption, (b) collection task. The information gain rate was higher in dynamic, compared to static environments, most notably in the consumption task or when worksite distance, D was small.

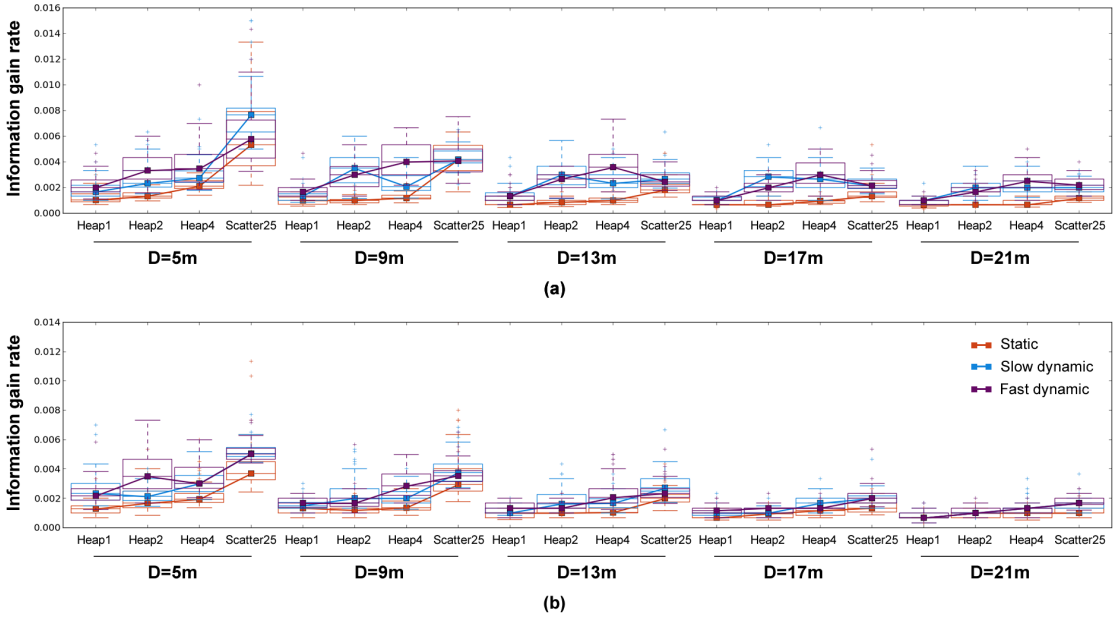


Figure E.12: Information gain rate, i, of 50-robot local broadcaster swarms in the (a) consumption, (b) collection task. The information gain rate was higher in some dynamic, compared to static environments, most notably in the consumption task or when worksite distance, D was small.

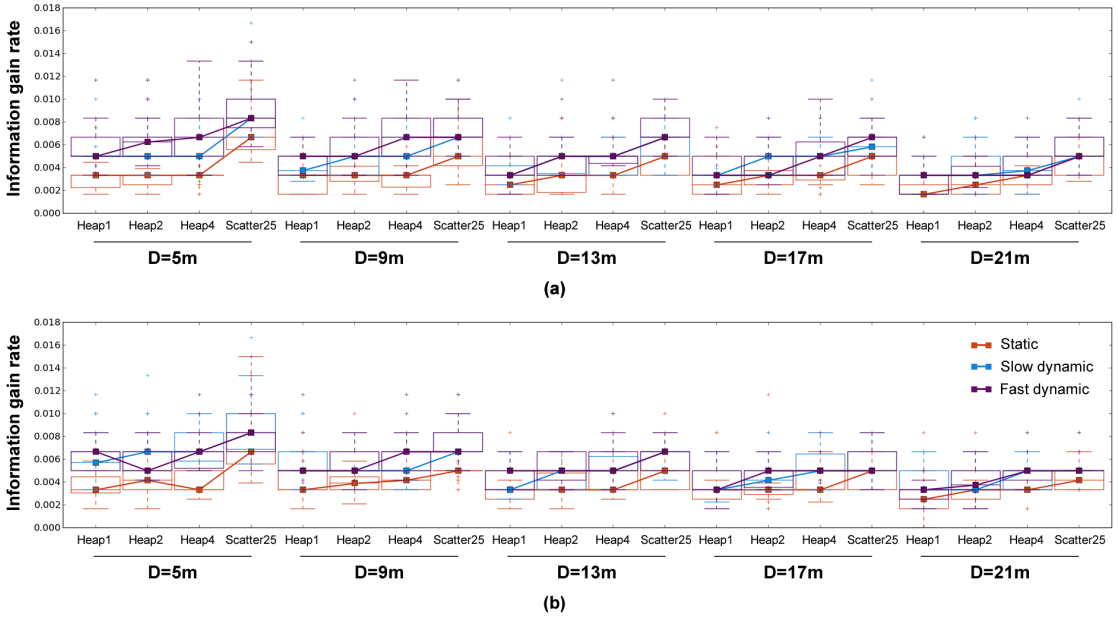


Figure E.13: Information gain rate, i, of 10-robot bee swarms in the (a) consumption, (b) collection task. The information gain rate was higher in a small number of dynamic, compared to static environments, most notably in the consumption task or when worksite distance, D was small.

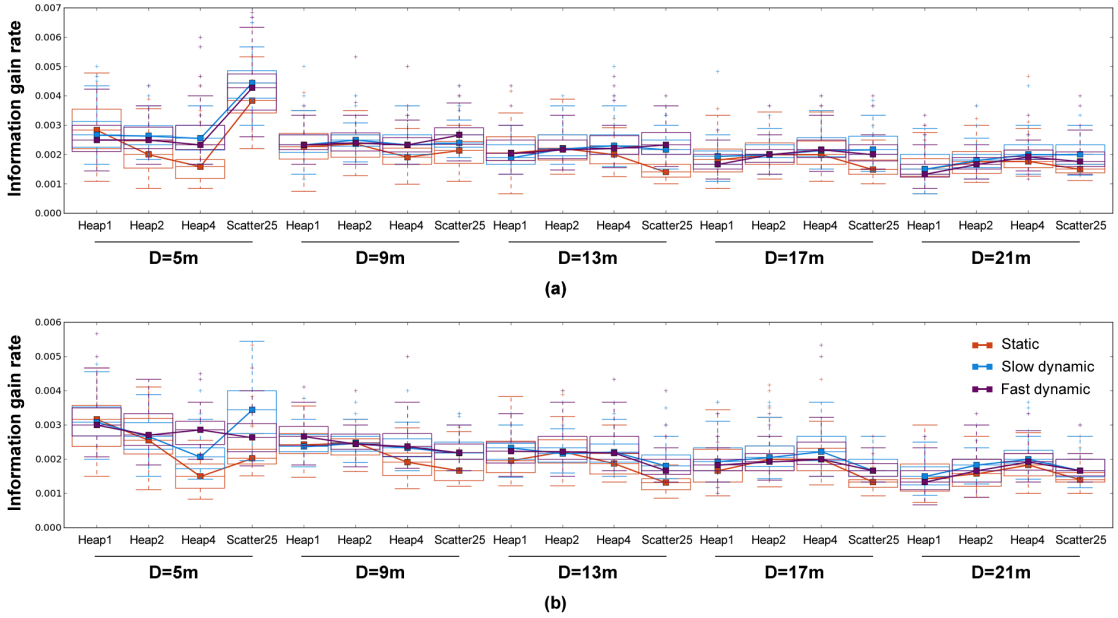


Figure E.14: Information gain rate, i, of 50-robot bee swarms in the (a) consumption, (b) collection task. The information gain rate was higher in a small number of dynamic, compared to static environments, most notably in the consumption task or when worksite distance, D was small.

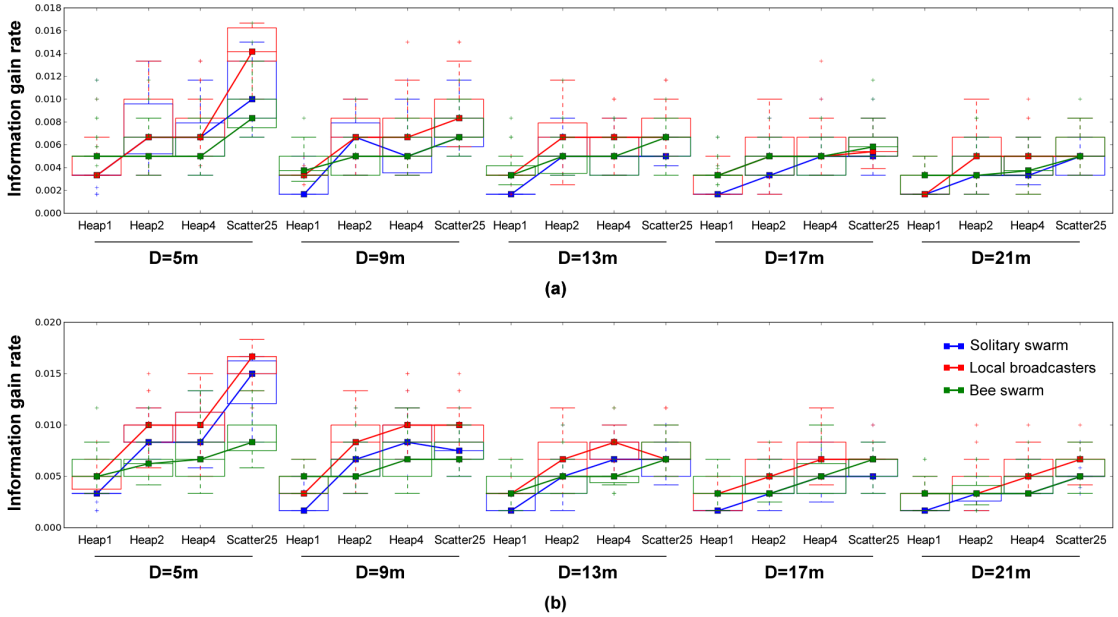


Figure E.15: Information gain rate, i, of 10-robot swarms in the (a) slow and (b) fast dynamic consumption task. Local broadcasters enjoyed the highest i in most scenarios. Bee swarms had the highest i in a small number of difficult scenarios, such as Heap1.

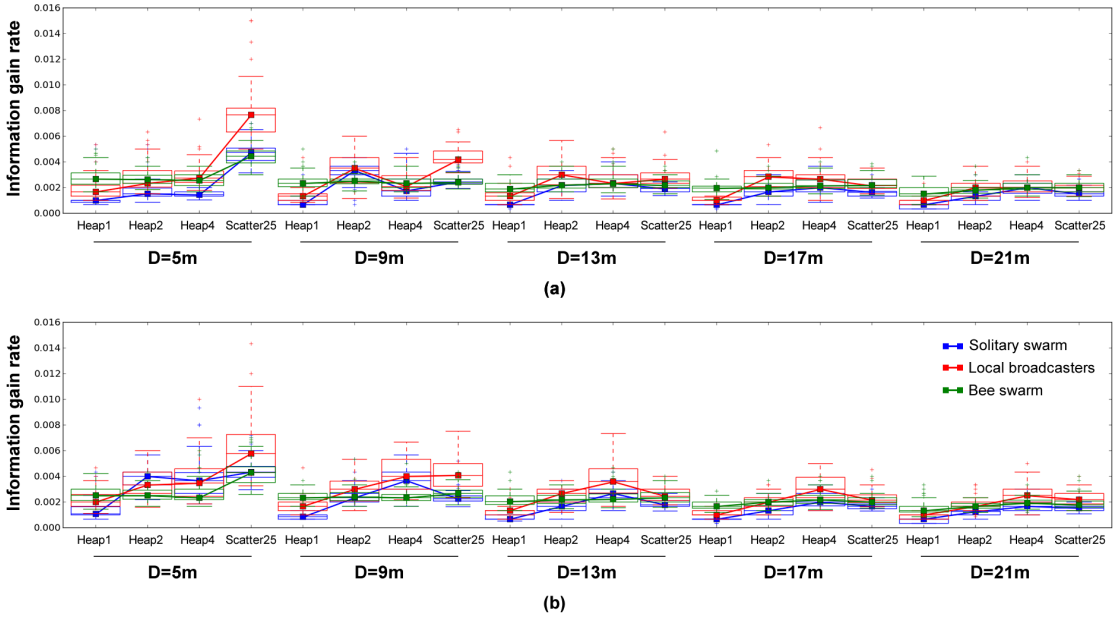


Figure E.16: Information gain rate, i, of 50-robot swarms in the (a) slow and (b) fast dynamic consumption task. Local broadcasters enjoyed the highest i in some scenarios with a large number of worksites and small D. Bee swarms had the highest i in a small number of difficult scenarios, such as Heap1.

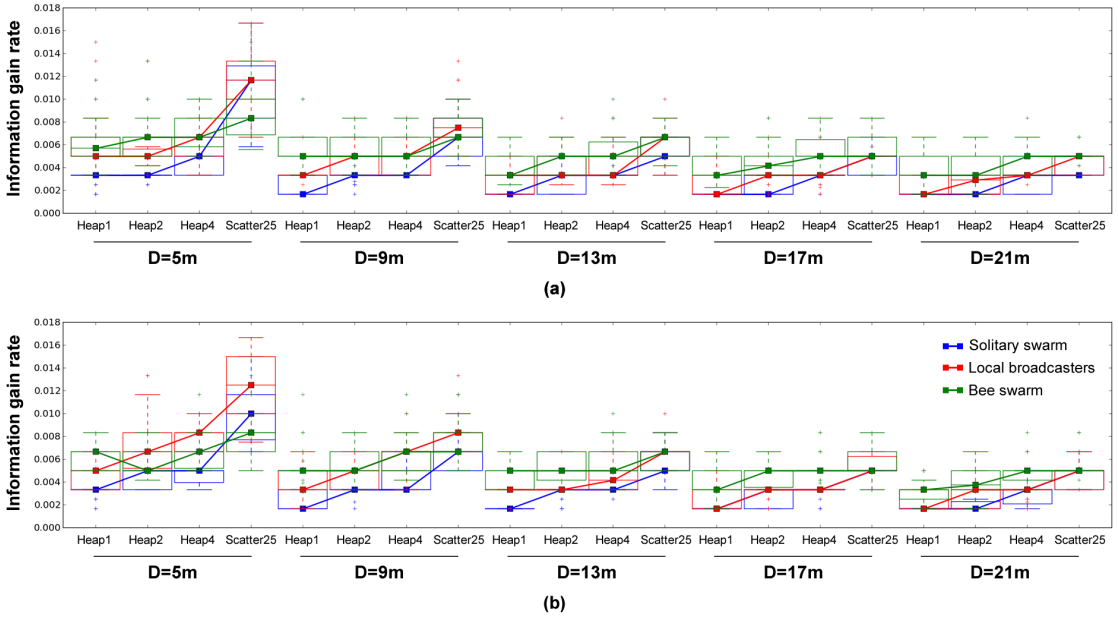


Figure E.17: Information gain rate, i, of 10-robot swarms in the (a) slow and (b) fast dynamic collection task. Bee swarms enjoyed the highest i in most scenarios. Local broadcasters had the highest i when the environmental change was fast, the number of worksites was large and when D was small.

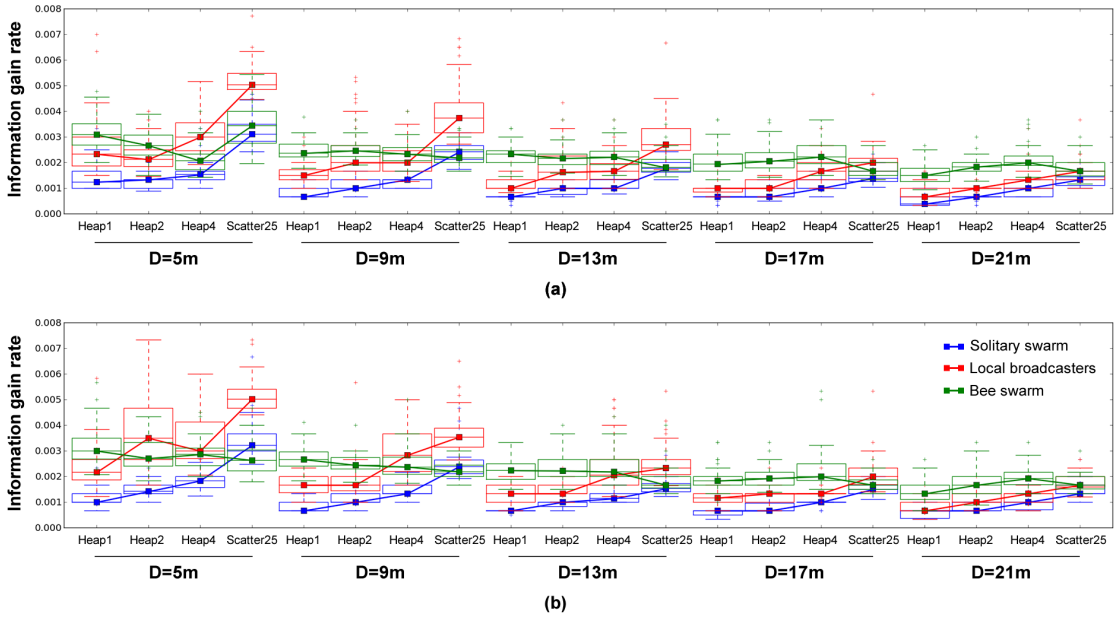


Figure E.18: Information gain rate, i, of 50-robot swarms in the (a) slow and (b) fast dynamic collection task. Bee swarms enjoyed the highest i in most scenarios. Local broadcasters had the highest i when the number of worksites was large or when D was small.

E.3 Misplacement cost coefficient

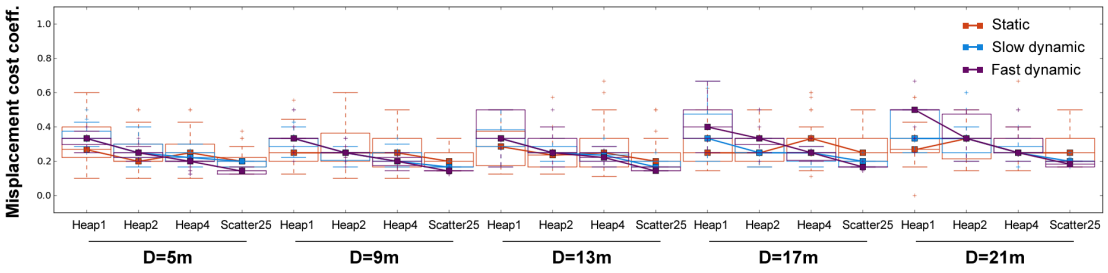


Figure E.19: Misplacement cost coefficient, m, of 10-robot local broadcaster swarms in the consumpion task. The value of m was similar in static and dynamic tasks in most environments.

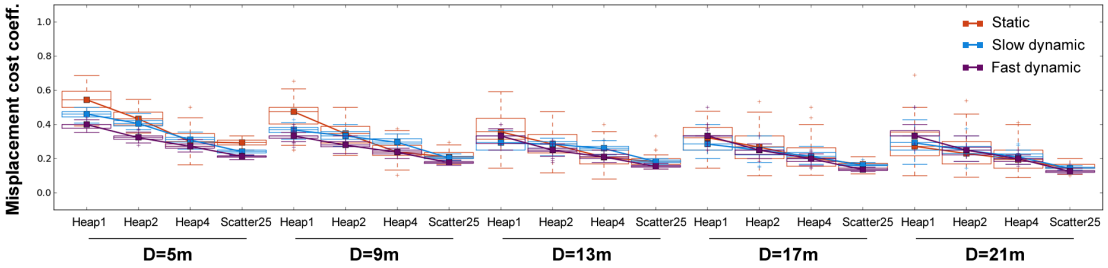


Figure E.20: Misplacement cost coefficient, m, of 50-robot local broadcaster swarms in the consumpion task. The value of m was similar in static and dynamic tasks in most environments. However, m was smaller in dynamic, compared to static, Heap1 scenarios when $D \leq 9$m.

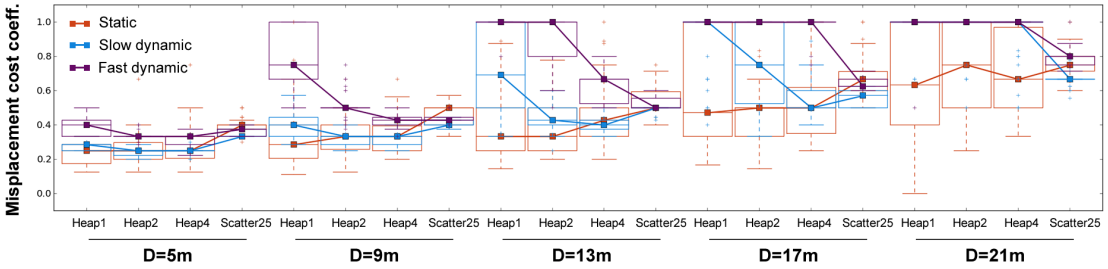


Figure E.21: Misplacement cost coefficient, m, of 10-robot bee swarms in the consumption task. The value of m was higher in dynamic, compared to static environments, most notably when worksite distance was large or when the environment changed quickly.

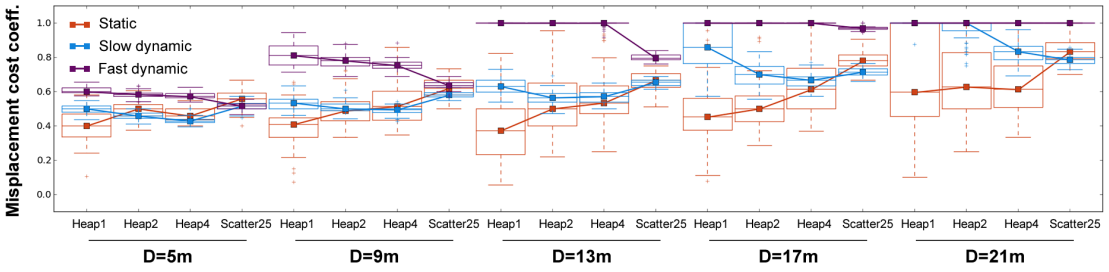


Figure E.22: Misplacement cost coefficient, m, of 10-robot bee swarms in the consumption task. The value of m was higher in dynamic, compared to static environments, most notably when worksite distance was large or when the environment changed quickly.

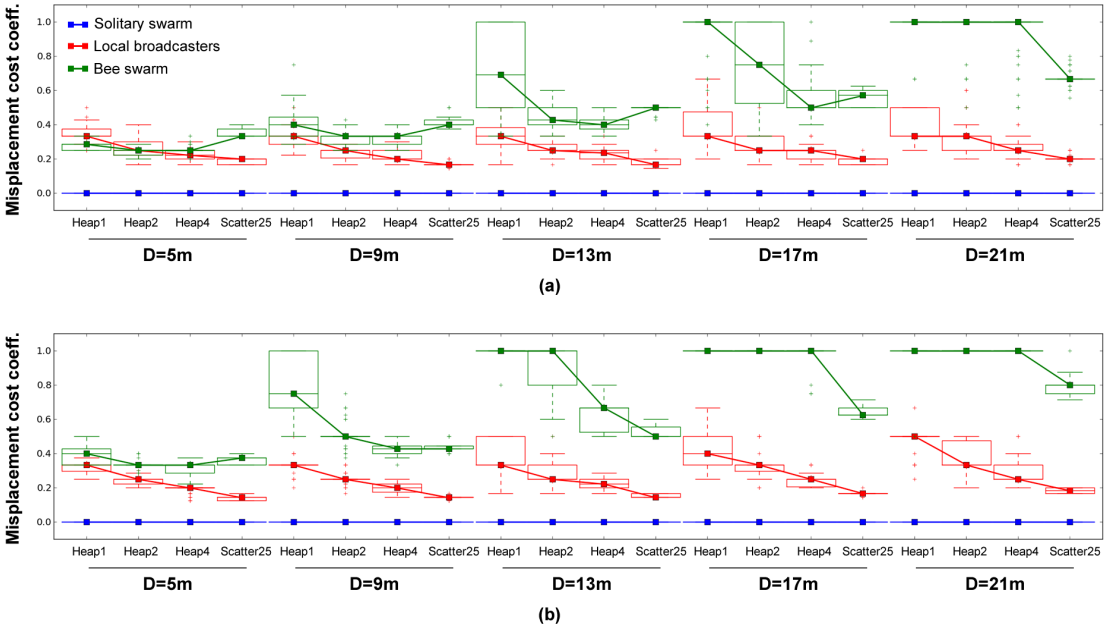


Figure E.23: Misplacement cost coefficient, m, of 10-robot swarms in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was the highest in most scenarios and reached the maximum value of 1 in the most difficult scenarios, especially when the environment changed quickly.

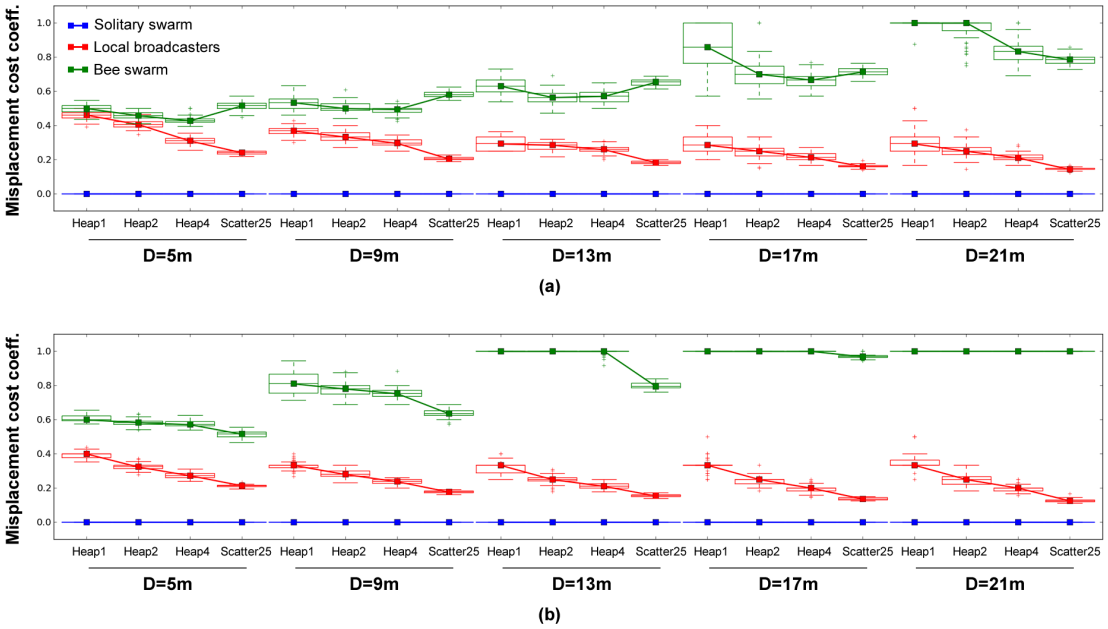


Figure E.24: Misplacement cost coefficient, m, of 50-robot swarms in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was the highest in most scenarios and reached the maximum value of 1 in the most difficult scenarios, especially when the environment changed quickly.

E.4 Opportunity cost

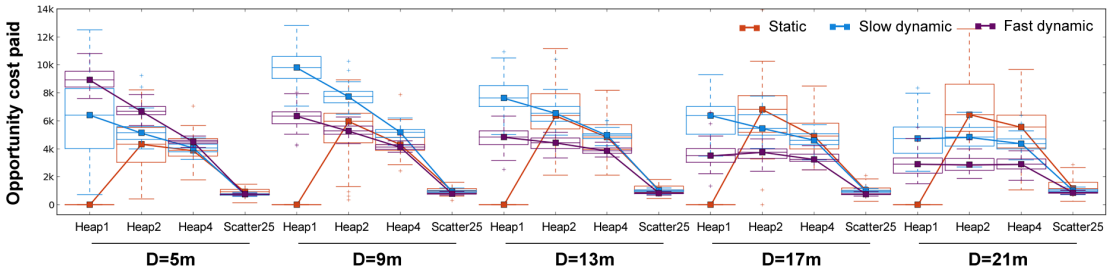


Figure E.25: Opportunity cost, C_O, of 10-robot solitary swarms in the collection task. The swarms paid a larger C_O in dynamic, compared to static, environments when D was small. The amount of C_O paid in dynamic environments was usually larger when the environment changed slowly.

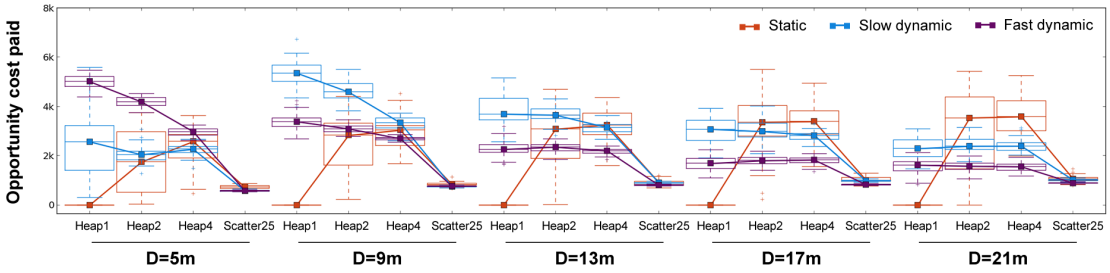


Figure E.26: Opportunity cost, C_O, of 50-robot solitary swarms in the collection task. The swarms paid a larger C_O in dynamic, compared to static, environments when D was small. The amount of C_O paid in dynamic environments was usually larger when the environment changed slowly.

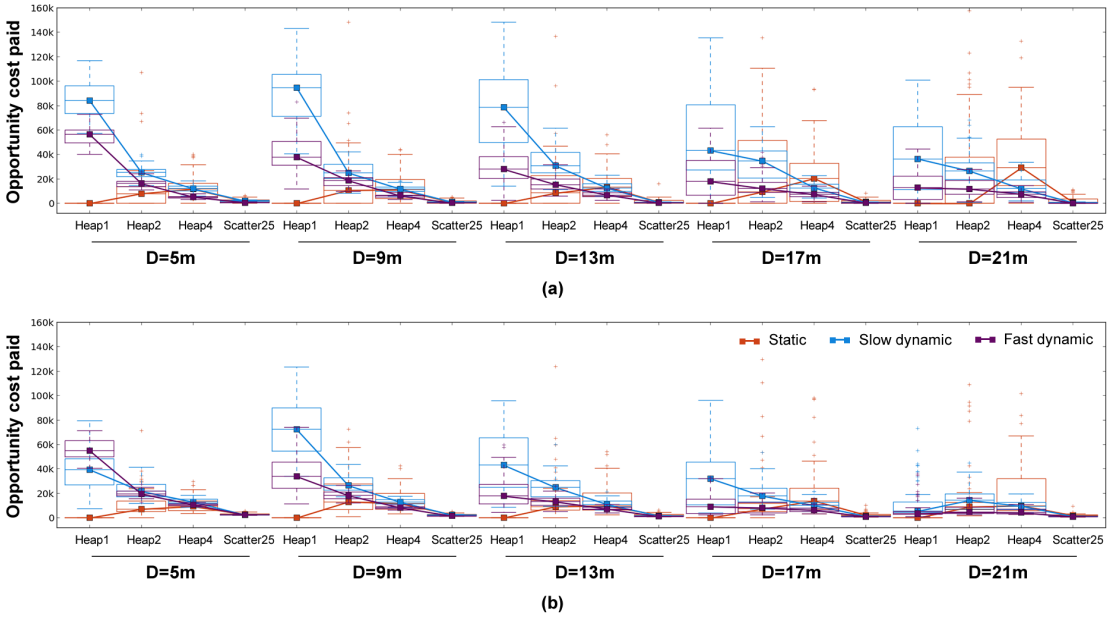


Figure E.27: Opportunity cost, C_O, of 10-robot local broadcaster swarms in the (a) consumption and (b) collection task. The amount of C_O paid was usually the largest in slow dynamic environments, especially when a few worksites were located close to the base, and the lowest in static environments.

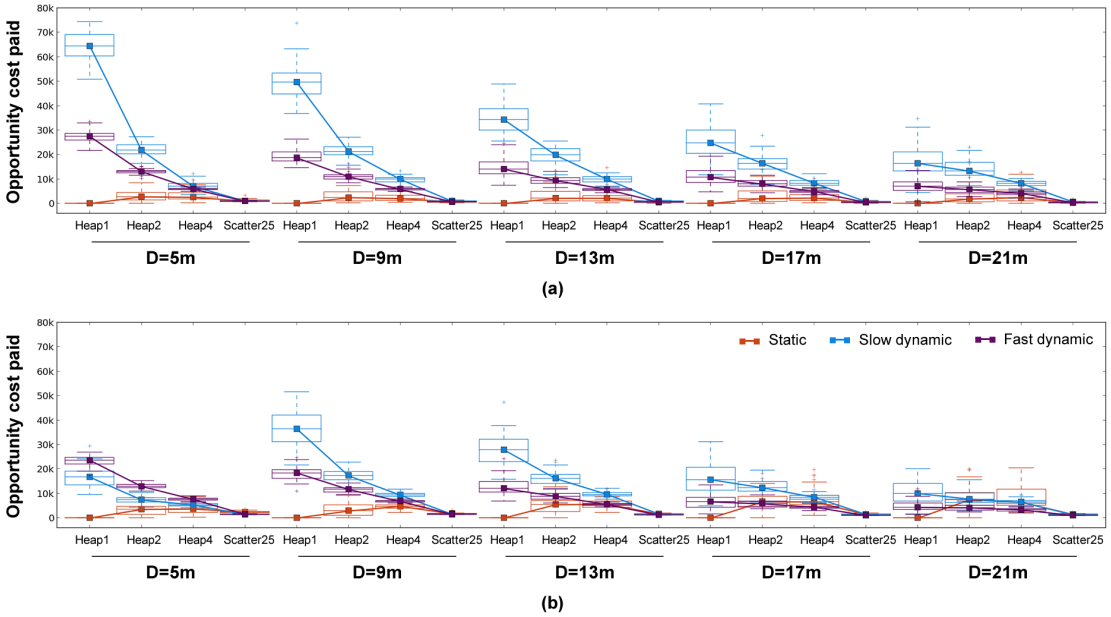


Figure E.28: Opportunity cost, C_O, of 50-robot local broadcaster swarms in the (a) consumption and (b) collection task. The amount of C_O paid was usually the largest in slow dynamic environments, especially when a few worksites were located close to the base, and the lowest in static environments.

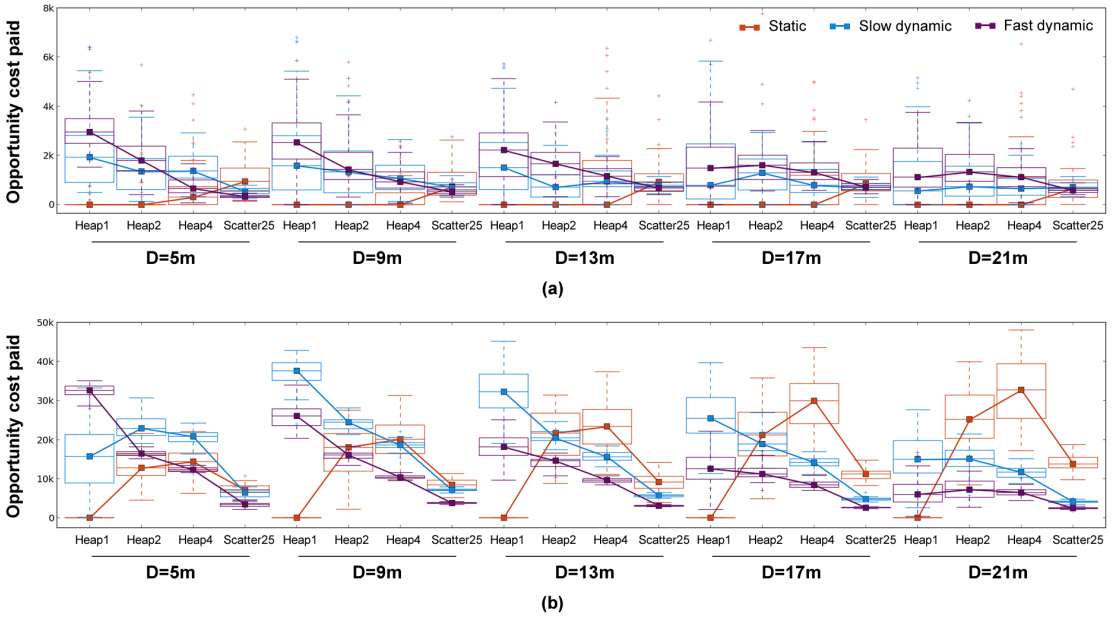


Figure E.29: Opportunity cost, C_O, of 10-robot bee swarms in the (a) consumption and (b) collection task. In the consumption task, the amount of C_O paid was usually the largest in the dynamic environments and when the environment changed quickly. In the collection task, the swarms paid a larger C_O in dynamic, compared to static, environments when D was small. The amount of C_O paid was usually larger in slow, compared to fast, dynamic environments.

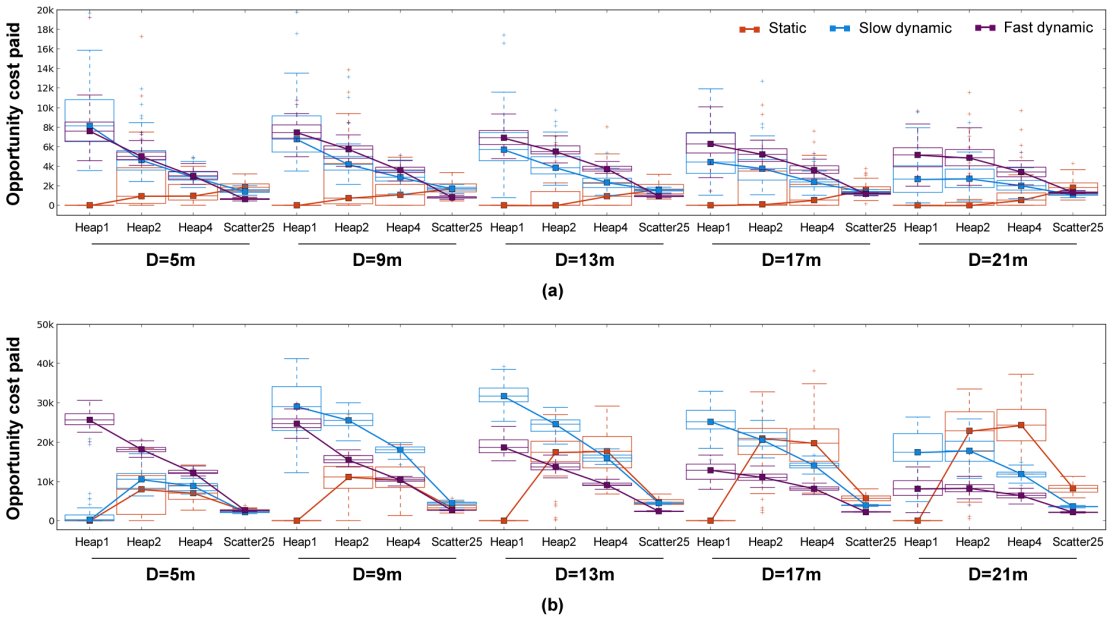


Figure E.30: Opportunity cost, C_O, of 50-robot bee swarms in the (a) consumption and (b) collection task. In the consumption task, the amount of C_O paid is usually the largest in the dynamic environments and when the environment changed quickly. In the collection task, the swarms paid a larger C_O in dynamic, compared to static, environments when D was small. The amount of C_O paid was usually larger in slow, compared to fast, dynamic environments.

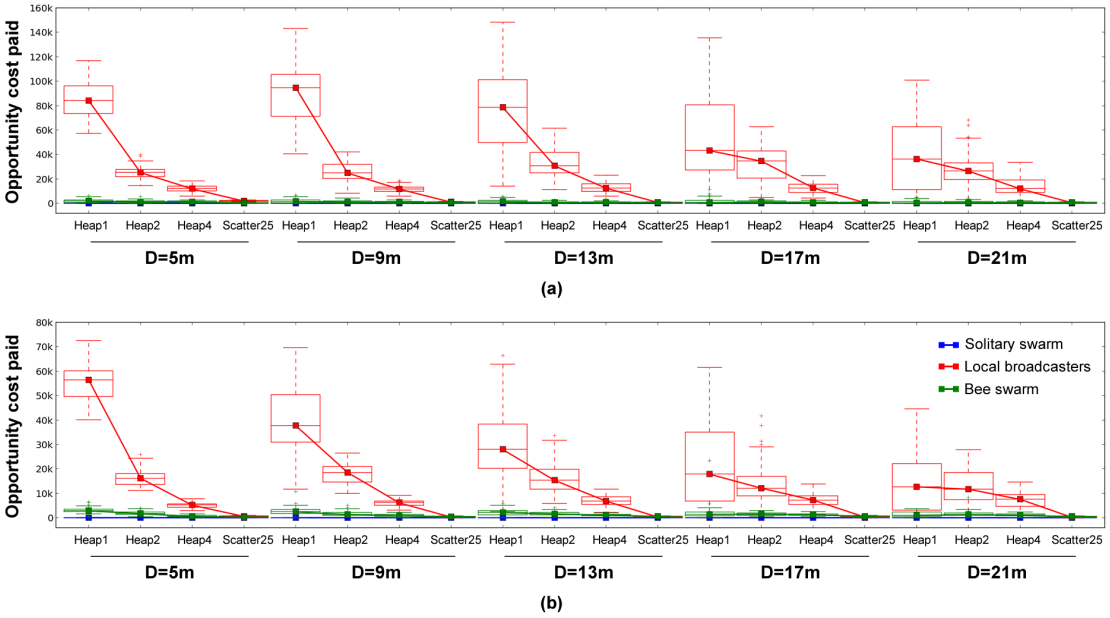


Figure E.31: Opportunity cost, C_O, of 10-robot swarms in the (a) slow and (b) fast dynamic consumption task. Local broadcasters paid the largest C_O in Heap environments, followed by bee and solitary swarms.

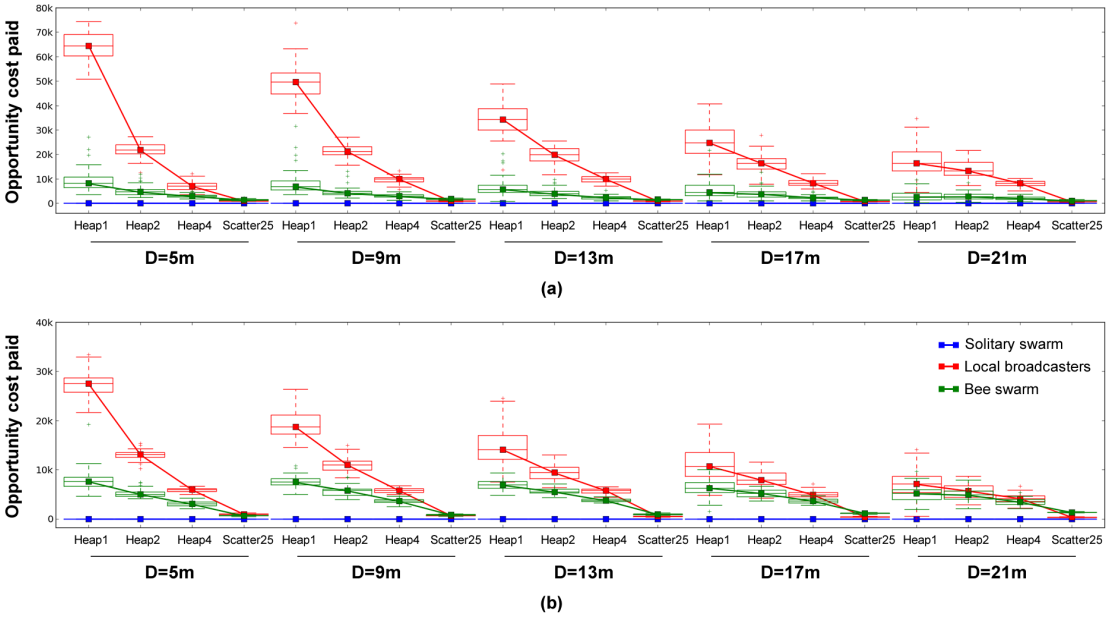


Figure E.32: Opportunity cost, C_O, of 50-robot swarms in the (a) slow and (b) fast dynamic consumption task. Local broadcasters paid the largest C_O in Heap environments, followed by bee and solitary swarms.

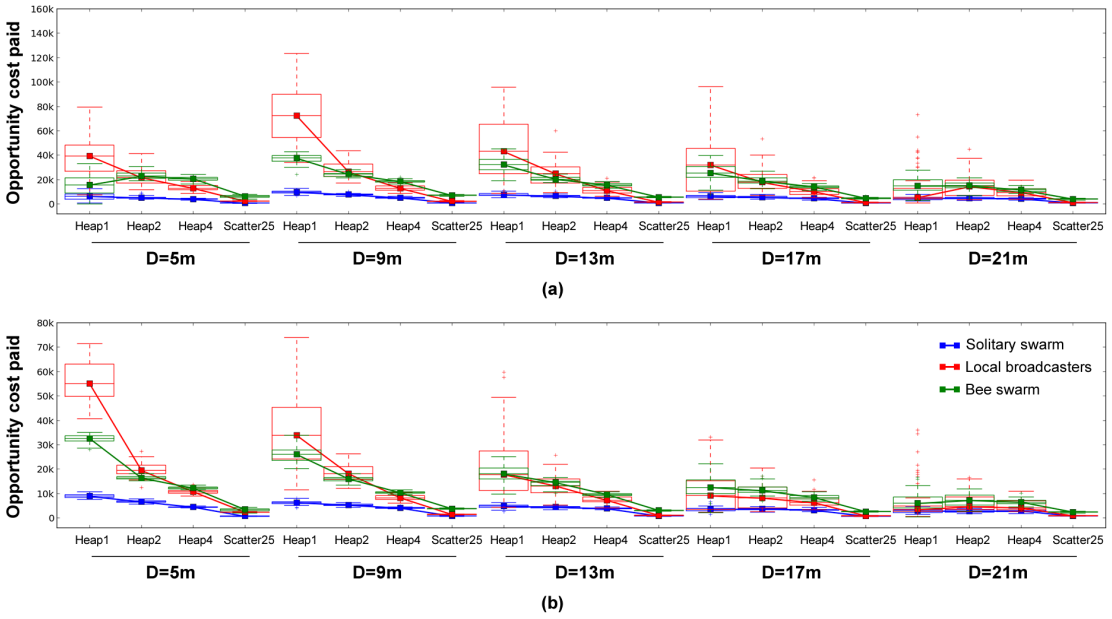


Figure E.33: Opportunity cost, C_O, of 10-robot swarms in the (a) slow and (b) fast dynamic collection task. Local broadcasters paid the largest C_O in Heap1 environments when D was small. Bee swarms and local broadcasters paid a similar amount of C_O in most of the other environments.

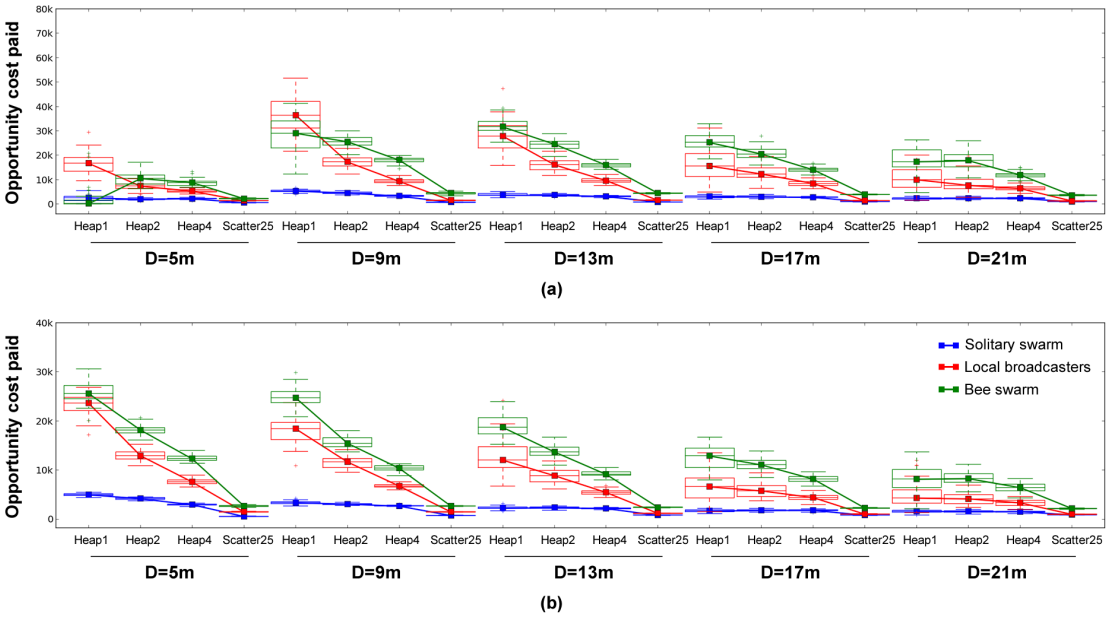


Figure E.34: Opportunity cost, C_O, of 50-robot swarms in the (a) slow and (b) fast dynamic collection task. Local broadcasters paid the largest C_O in Heap1 environments when D was small. Bee swarms paid the largest C_O in most of the other environments.

E.5 Reward obtained

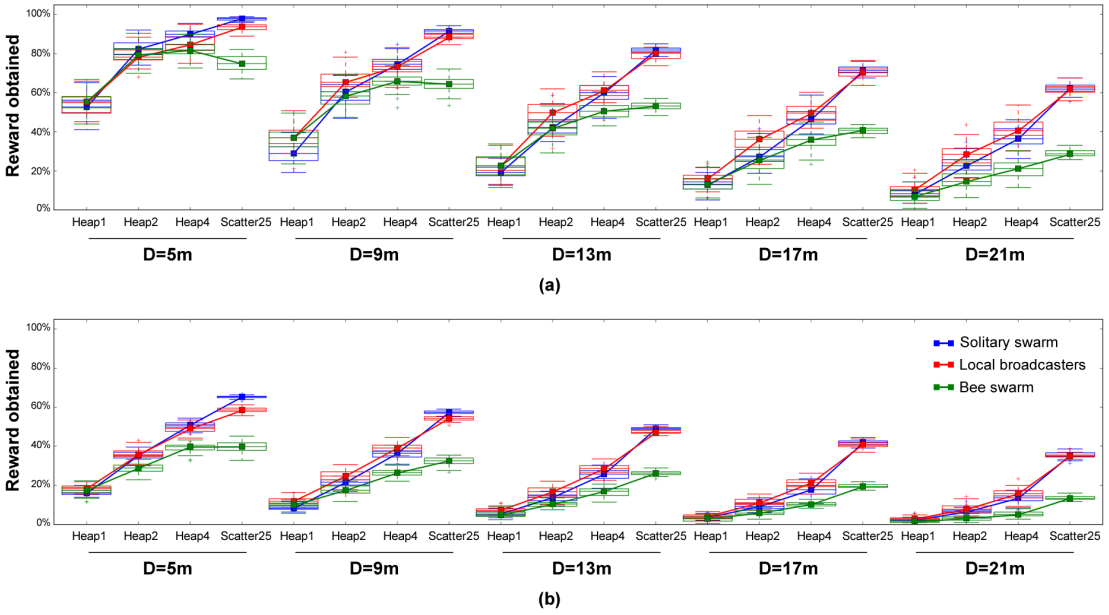


Figure E.35: The percentage of available reward obtained by 10-robot swarms in the (a) slow and (b) fast dynamic consumption task. Solitary swarms outperformed the other swarms when D was small, especially in Scatter25 environments. The performance of bee swarms was usually worse than that of the other swarms.

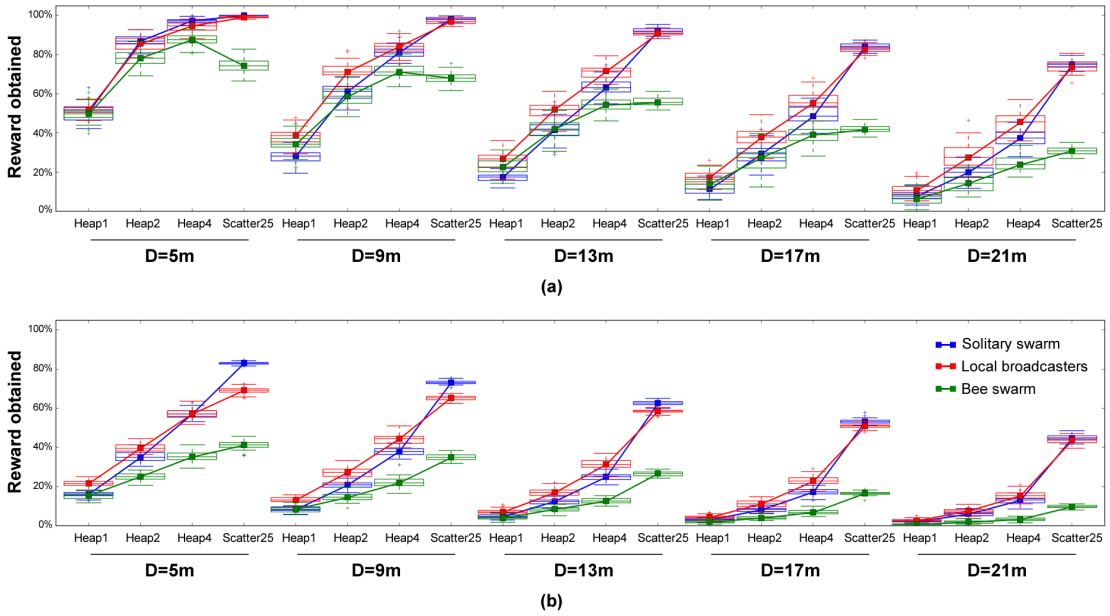


Figure E.36: The percentage of available reward obtained by 25-robot swarms in the (a) slow and (b) fast dynamic consumption task. Solitary swarms outperformed the other swarms in Scatter25 environments, especially when the environment changed quickly. The performance of bee swarms was always worse than that of the other swarms.

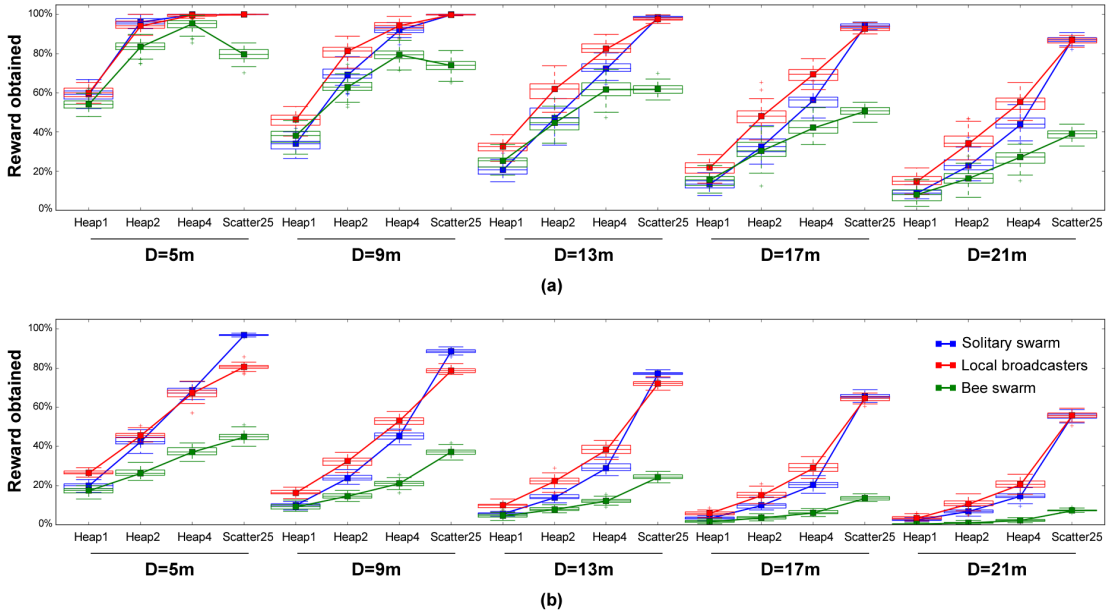


Figure E.37: The percentage of available reward obtained by 50-robot swarms in the (a) slow and (b) fast dynamic consumption task. Solitary swarms outperformed the other swarms in Scatter25 environments with a small D, especially when the environment changed quickly. The performance of bee swarms was always worse than that of the other swarms.

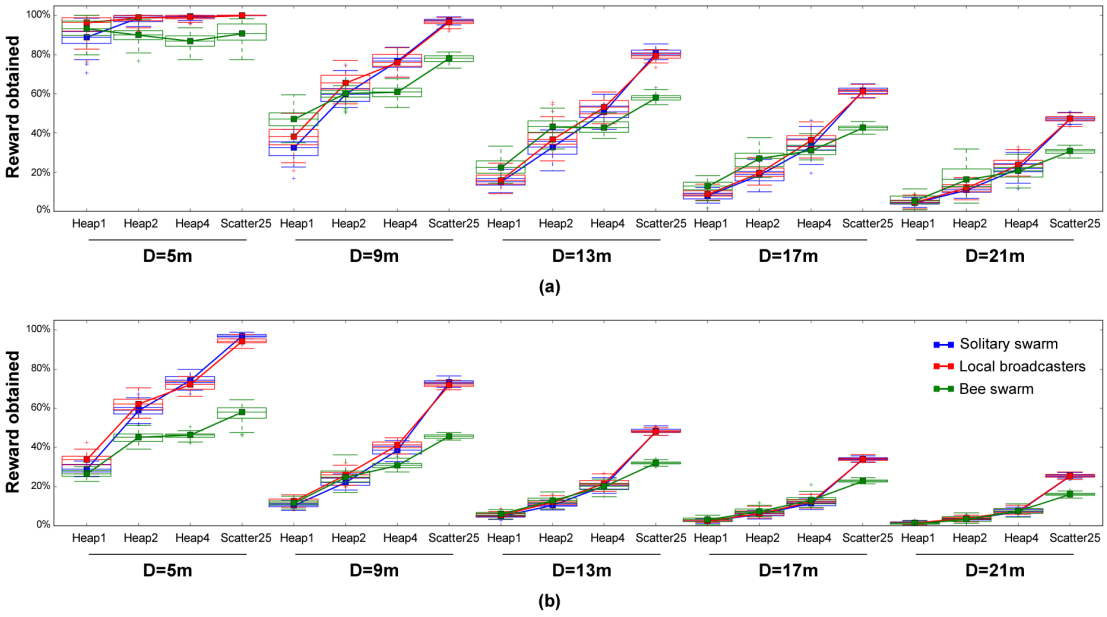


Figure E.38: The percentage of available reward obtained by 10-robot swarms in the (a) slow and (b) fast dynamic collection task. Solitary swarms outperformed the other Swarms in Scatter25 environments with a small D when the environment changed quickly. The performance of bee swarms was better than that of the other swarms in most Heap1 and Heap2 environments when the environment changed slowly.

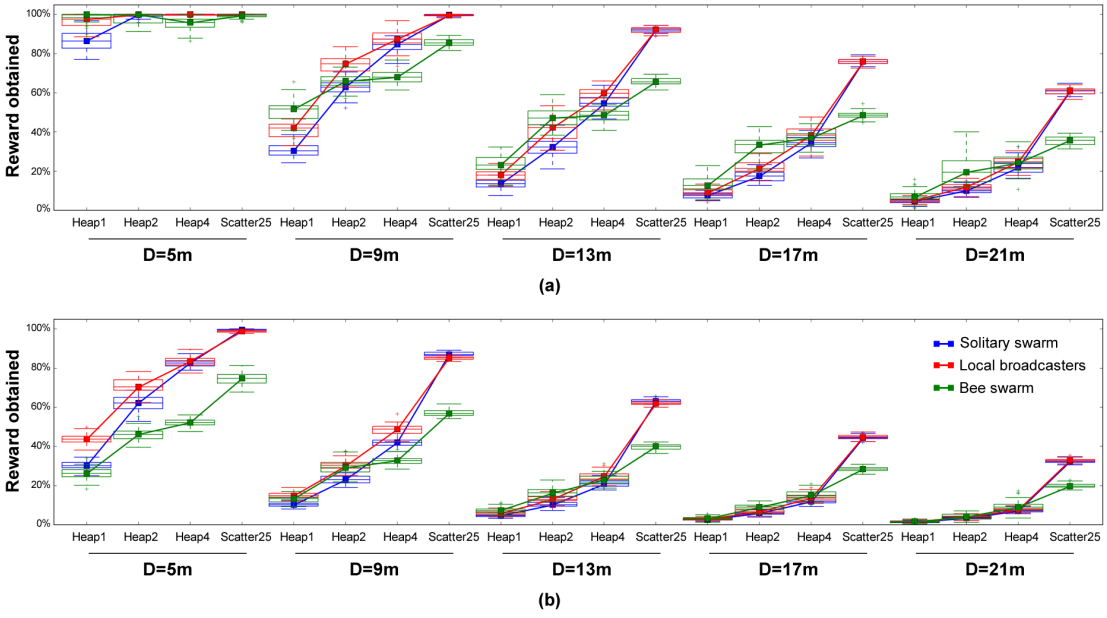


Figure E.39: The percentage of available reward obtained by 25-robot swarms in the (a) slow and (b) fast dynamic collection task. Solitary swarms outperformed the other Swarms in Scatter25 environments with 9m $\leq D \geq$ 13m when the environment changed quickly. The performance of bee swarms was better than that of the other swarms in most Heap1 and Heap2 environments, most notably when the environment changed slowly.

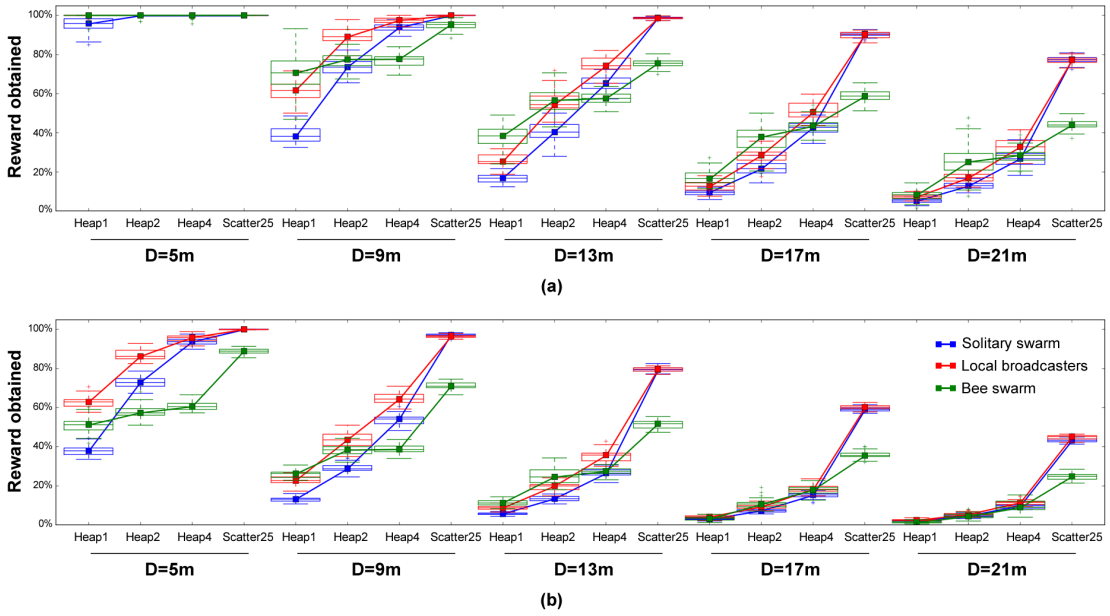


Figure E.40: The percentage of available reward obtained by 50-robot swarms in the (a) slow and (b) fast dynamic collection task. Local broadcasters outperformed the other swarms in most environments. The performance of bee swarms was better than that of the other swarms in most Heap1 and Heap2 environments, most notably when the environment changed slowly.

Appendix F

Supporting graphs for the analysis of the Opportunism add-on strategy

F.1 Information gain rate

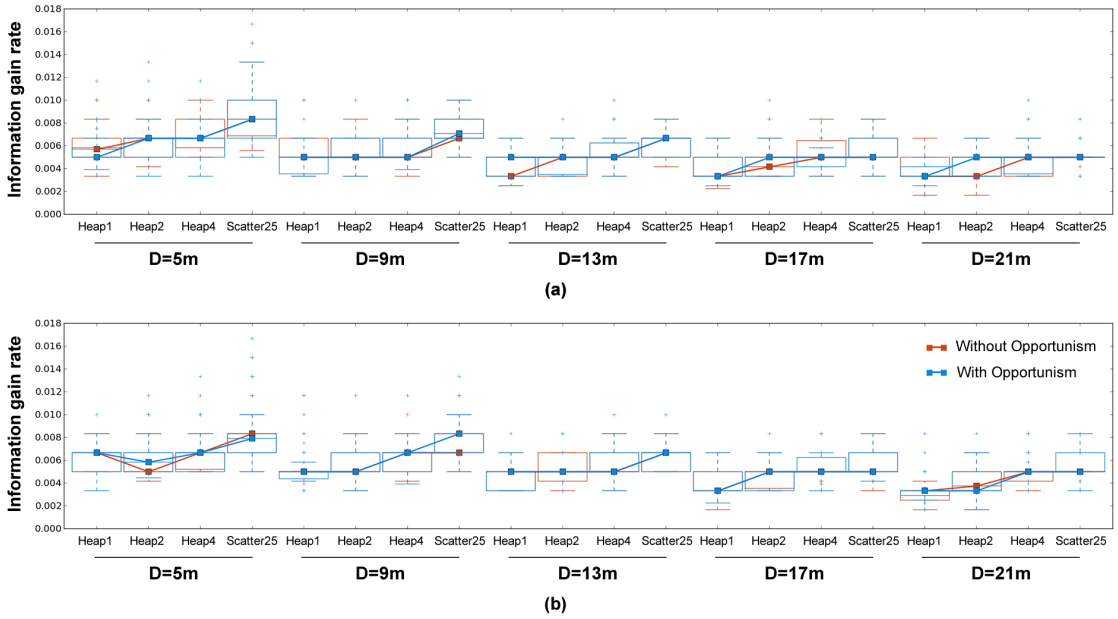


Figure F.1: Information gain rate, i, of 10-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. Opportunism did not affect the i of bee swarms.

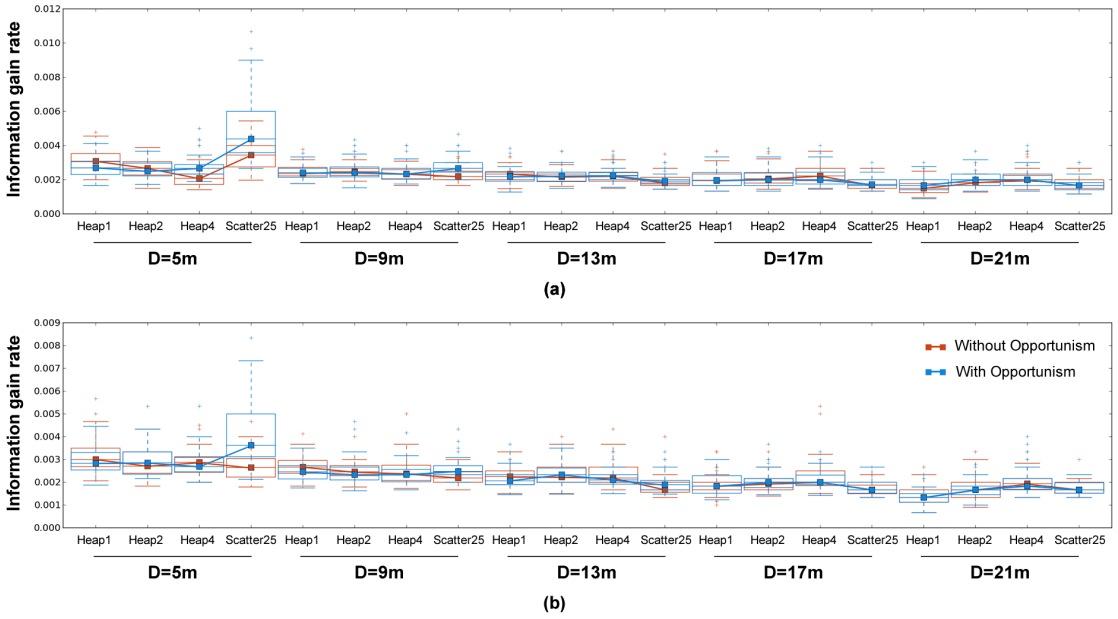


Figure F.2: Information gain rate, i, of 50-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The i of bee swarms was higher with Opportunism in the Scatter25 scenarios with $D = 5$m.

F.2 Misplacement cost coefficient

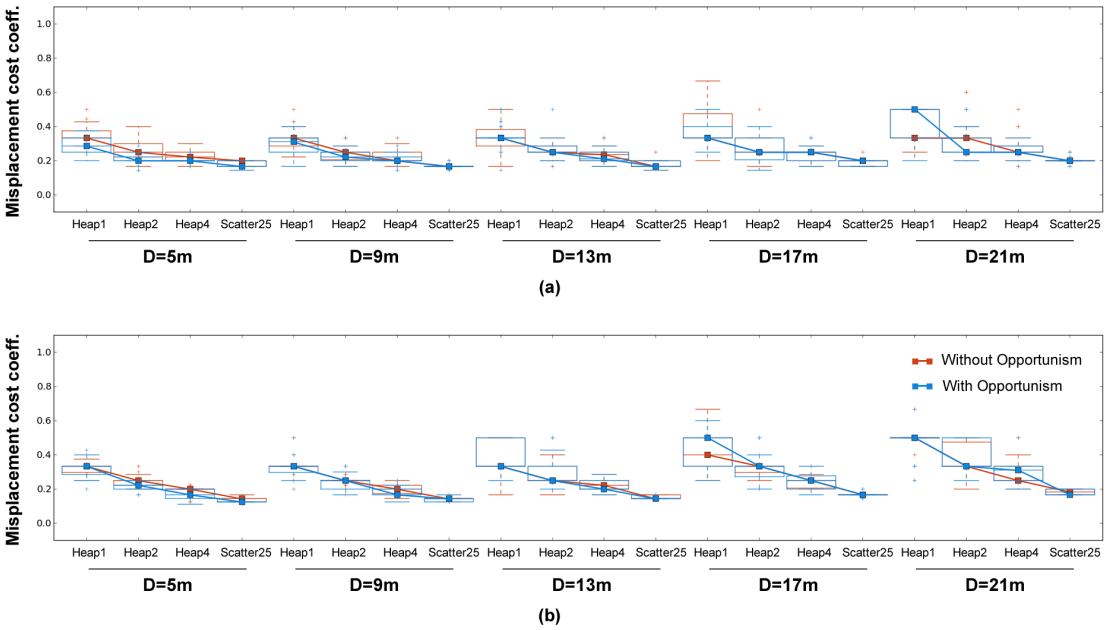


Figure F.3: Misplacement cost coefficient, m, of 10-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The m of local broadcasters was smaller with Opportunism in some scenarios when $D = 5$m.

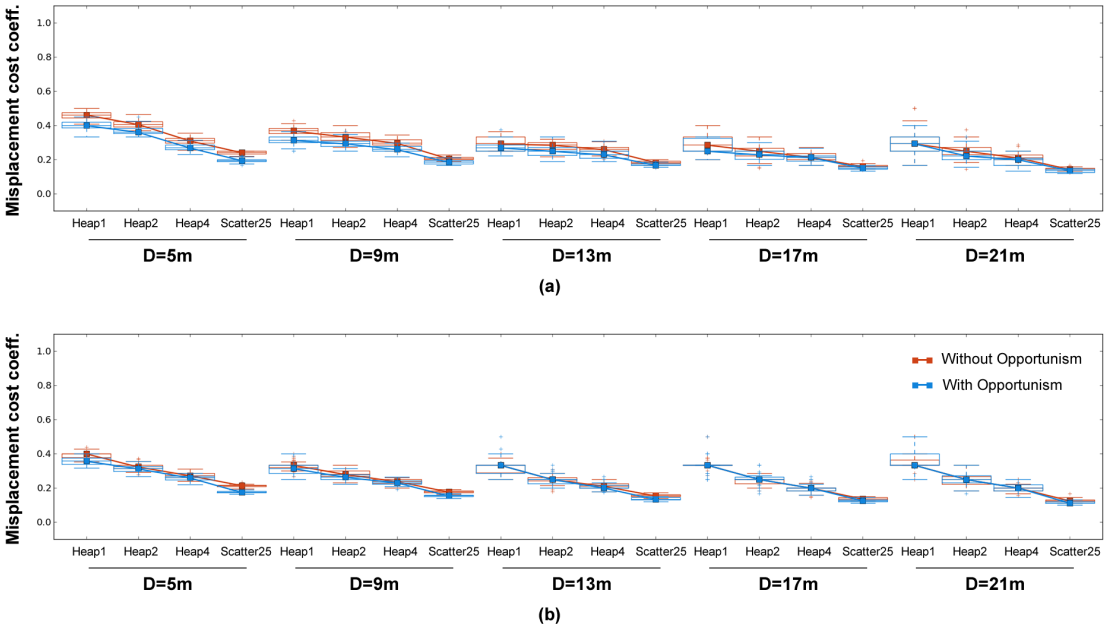


Figure F.4: Misplacement cost coefficient, m, of 50-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The m of local broadcasters was smaller with Opportunism when D was small, especially when the environment changed slowly.

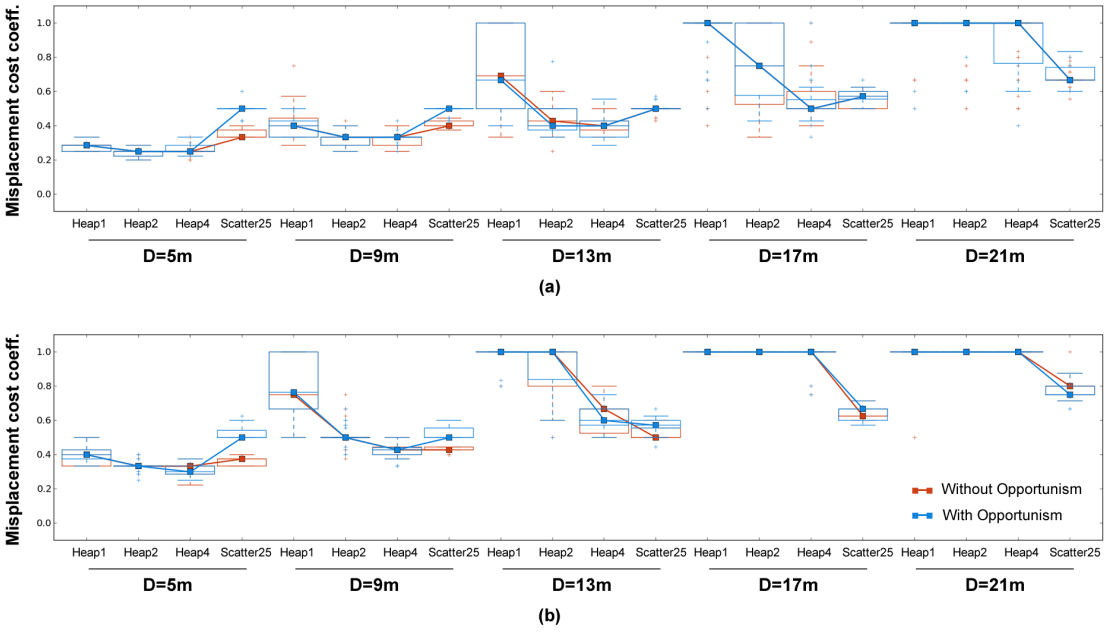


Figure F.5: Misplacement cost coefficient, m, of 10-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was larger with Opportunism in Scatter25 scenarios when D was small.

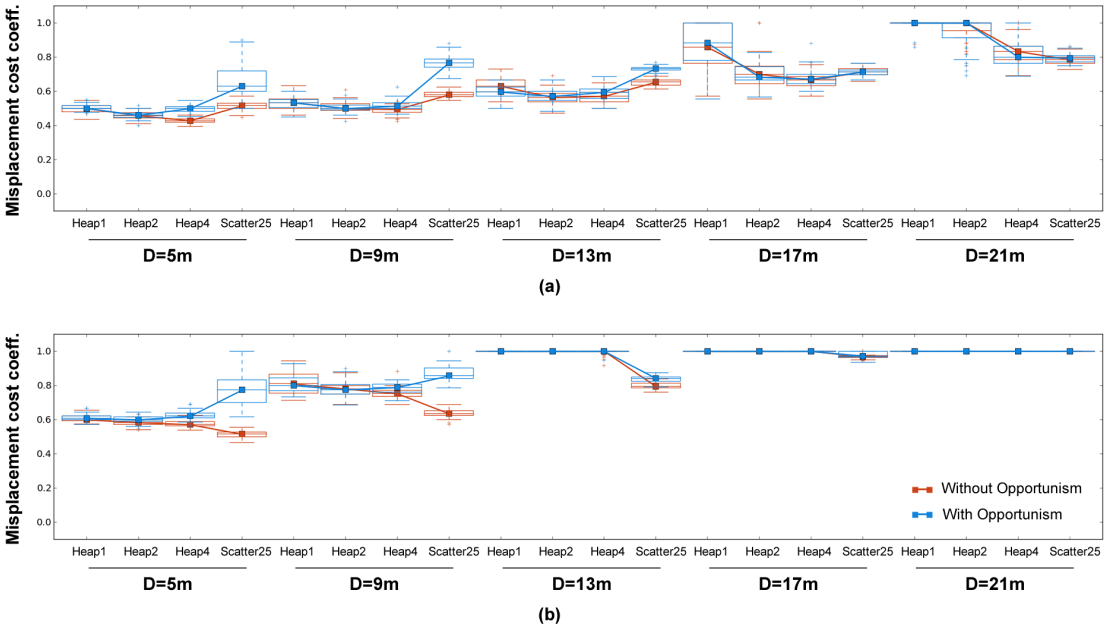


Figure F.6: Misplacement cost coefficient, m, of 50-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was larger with Opportunism in Heap4 and Scatter25 scenarios when D was small.

F.3 Opportunity cost

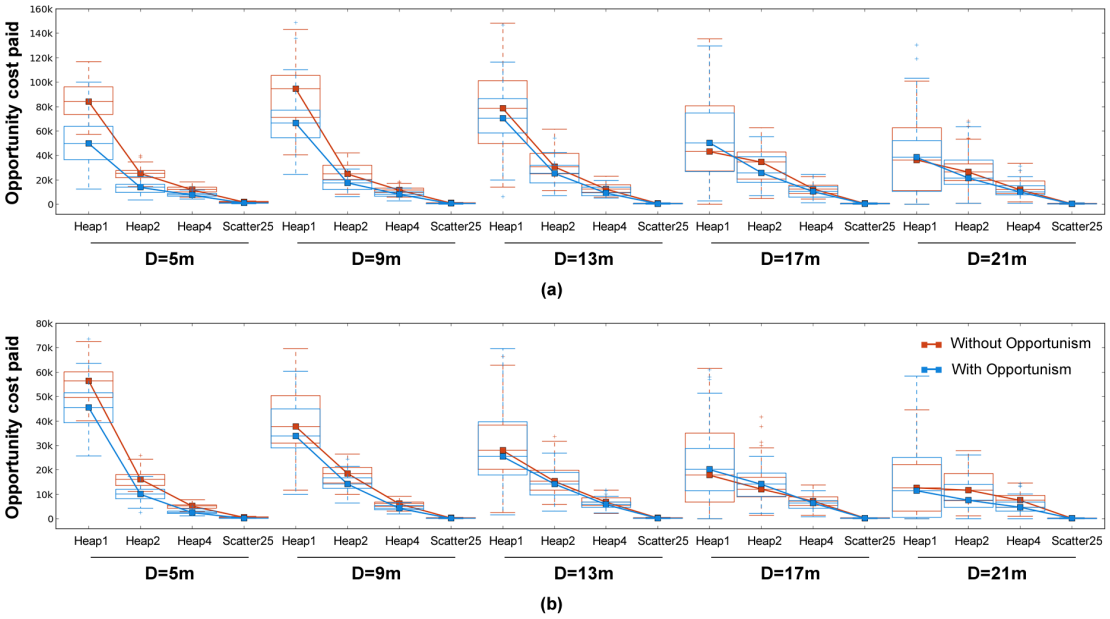


Figure F.7: Opportunity cost, C_O, of 10-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The C_O paid by local broadcasters was smaller with Opportunism in Heap scenarios, especially when D was small and when the environment changed slowly.

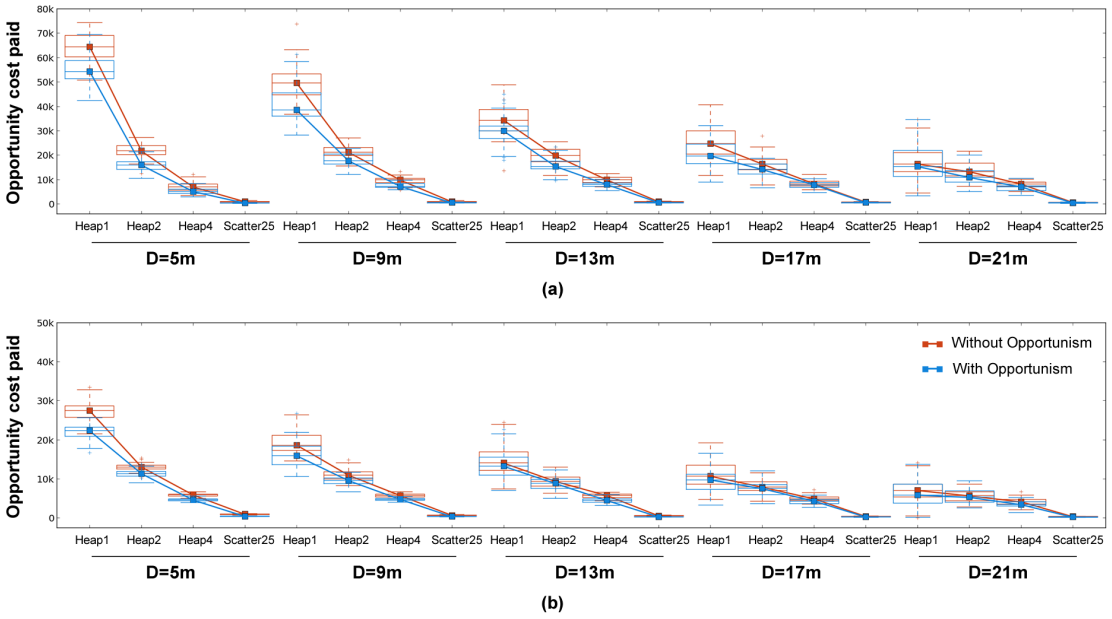


Figure F.8: Opportunity cost, C_O, of 50-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The C_O paid by local broadcasters was smaller with Opportunism in some Heap scenarios, especially when D was small and when the environment changed slowly.

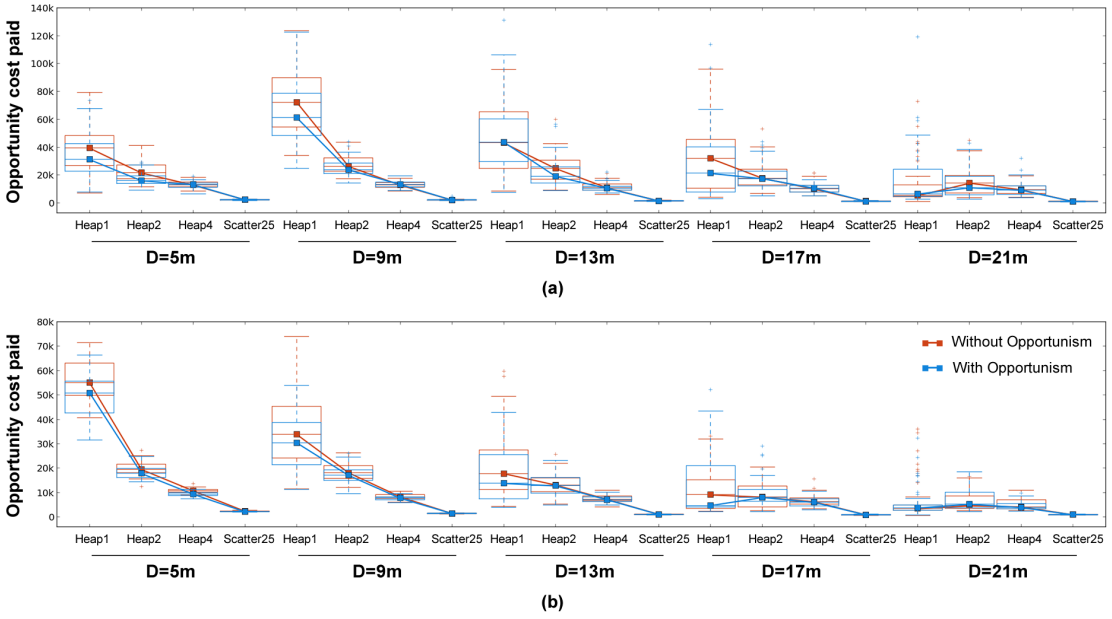


Figure F.9: Opportunity cost, C_O, of 10-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The C_O paid by local broadcasters was smaller with Opportunism in some Heap1 scenarios.

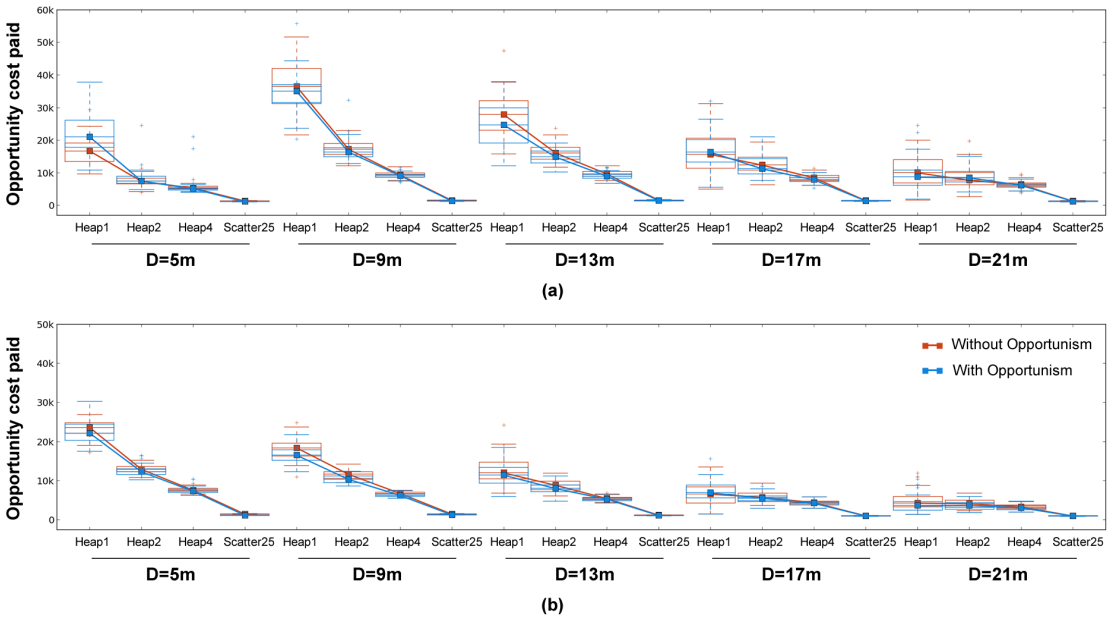


Figure F.10: Opportunity cost, C_O, of 50-robot local broadcaster swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The C_O paid by local broadcasters was mostly not affected by Opportunism.

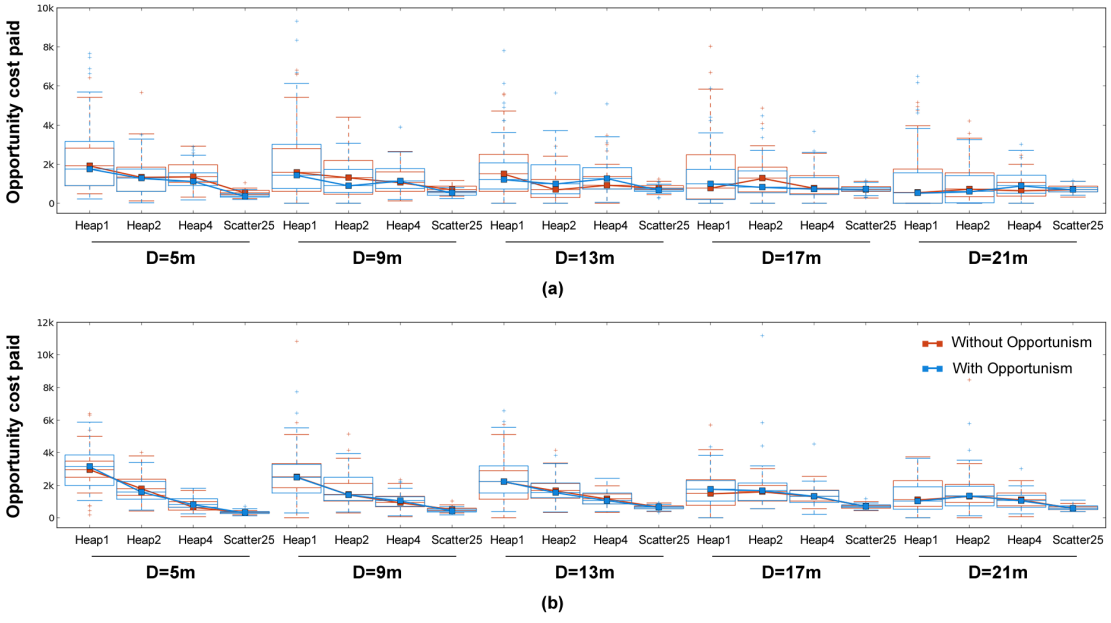


Figure F.11: Opportunity cost, C_O, of 10-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by bee swarms was not affected by Opportunism in most of the scenarios.

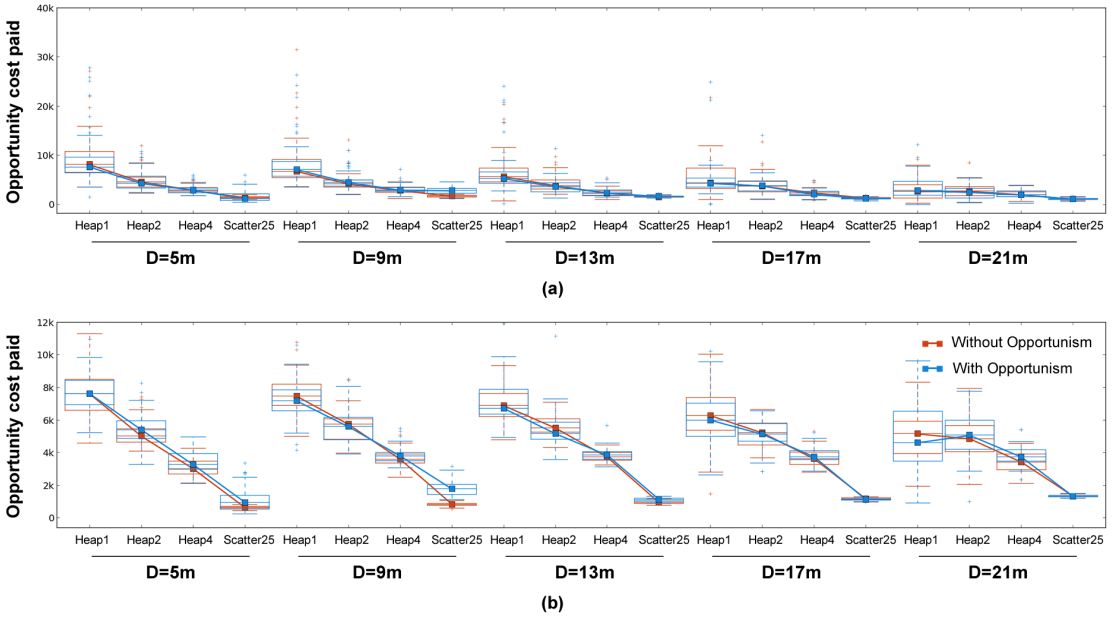


Figure F.12: Opportunity cost, C_O, of 50-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by bee swarms was not affected by Opportunism in most of the scenarios.

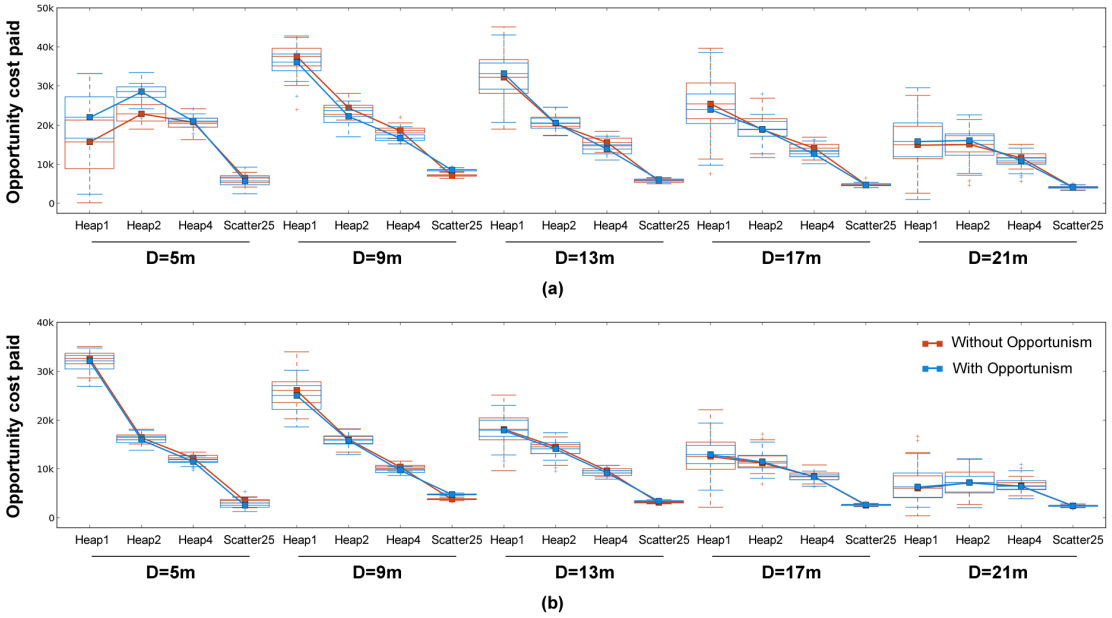


Figure F.13: Opportunity cost, C_O, of 10-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by bee swarms was larger with Opportunism in Heap2 and Heap4 scenarios when $D = 5$m and when the environment changed slowly. The amount of C_O paid was smaller with Opportunism in Heap2 and Heap4 scenarios when $D = 9$m and when the environment changed slowly.

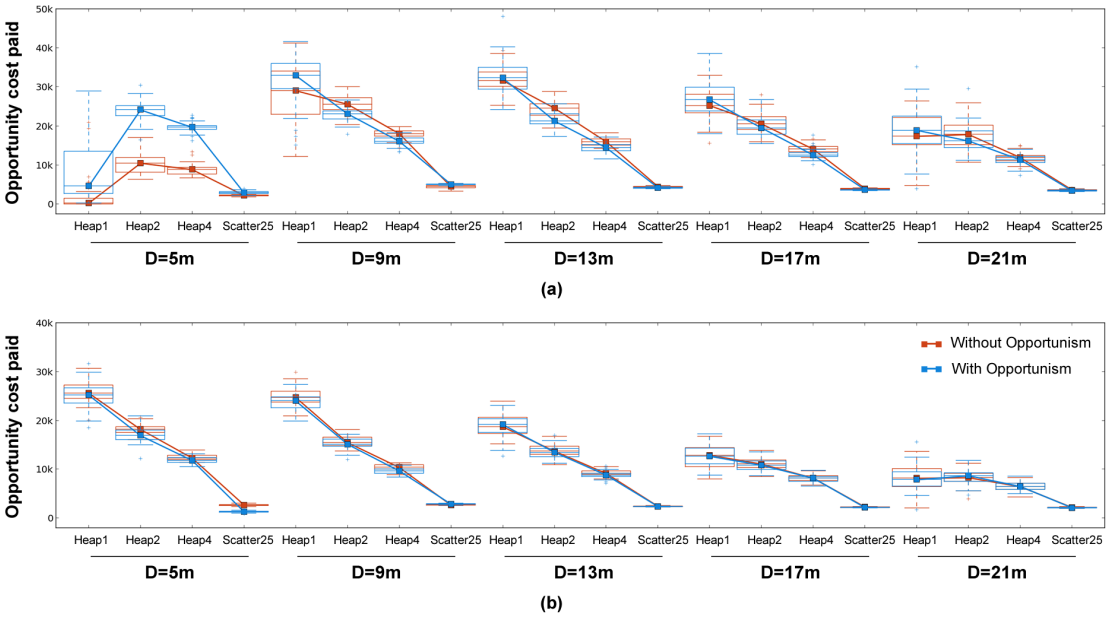


Figure F.14: Opportunity cost, C_O, of 50-robot bee swarms with and without Opportunism in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by bee swarms was larger with Opportunism in Heap scenarios when $D = 5$m and when the environment changed slowly. The amount of C_O paid was smaller with Opportunism in some Heap2 and Heap4 scenarios when $D > 5$m, especially when the environment changed slowly.

F.4 Reward obtained

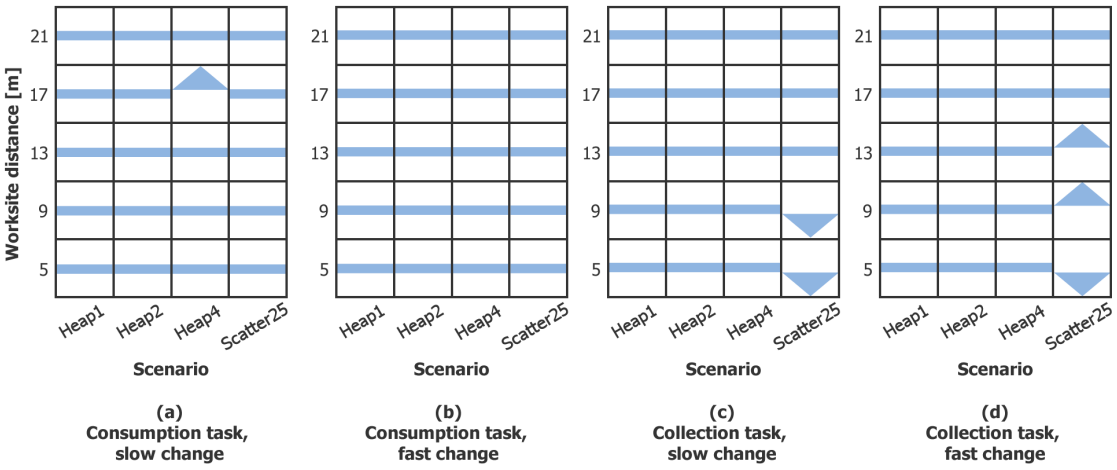


Figure F.15: The effect of Opportunism on the performance of 10-robot solitary swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure F.21.

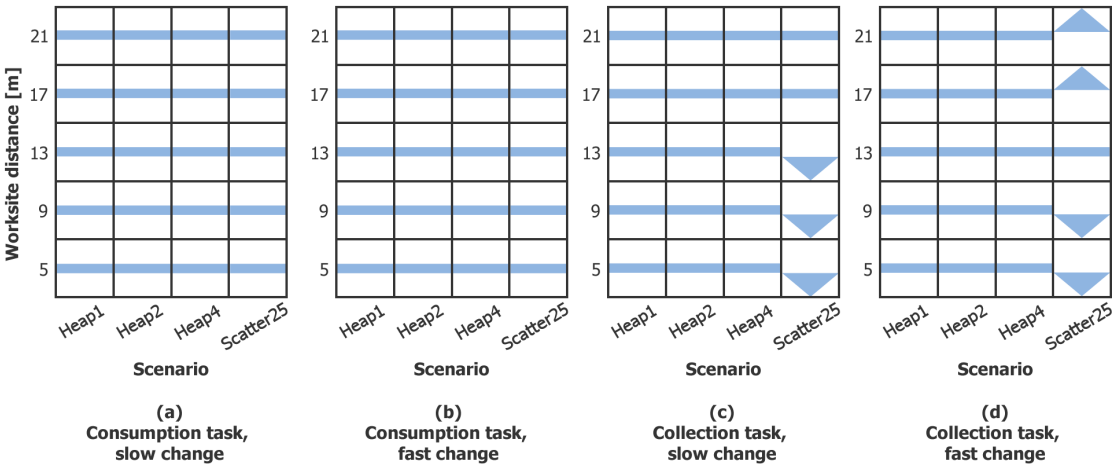


Figure F.16: The effect of Opportunism on the performance of 50-robot solitary swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure F.23.

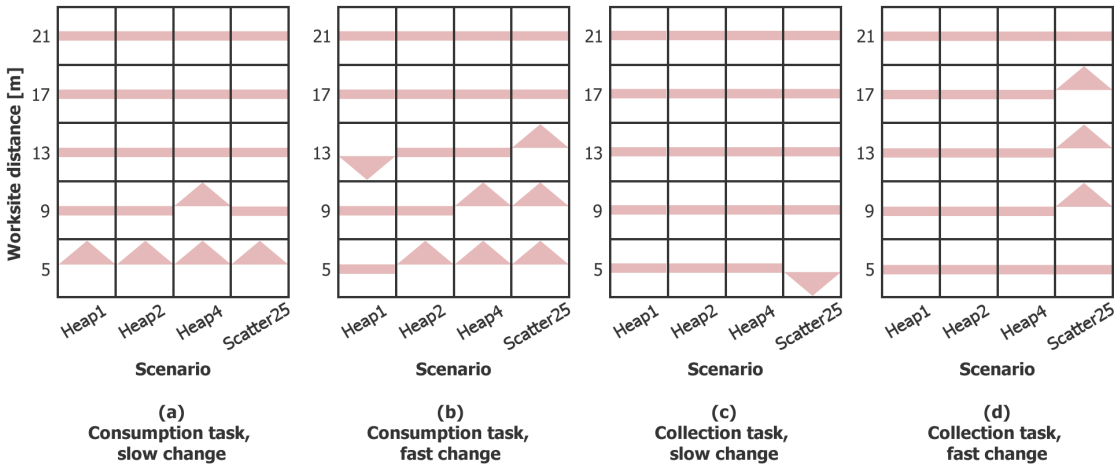


Figure F.17: The effect of Opportunism on the performance of 10-robot local broadcaster swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure F.24.

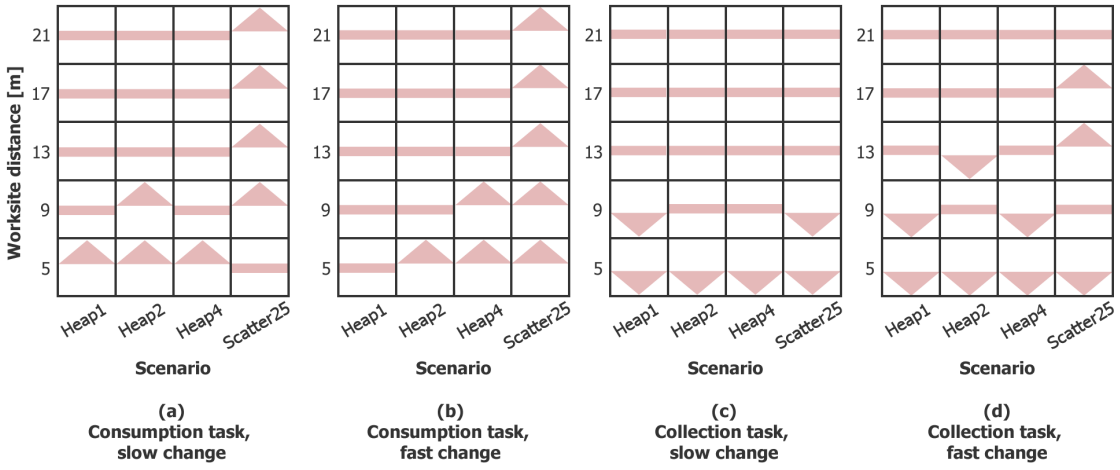


Figure F.18: The effect of Opportunism on the performance of 50-robot local broadcaster swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure F.26.

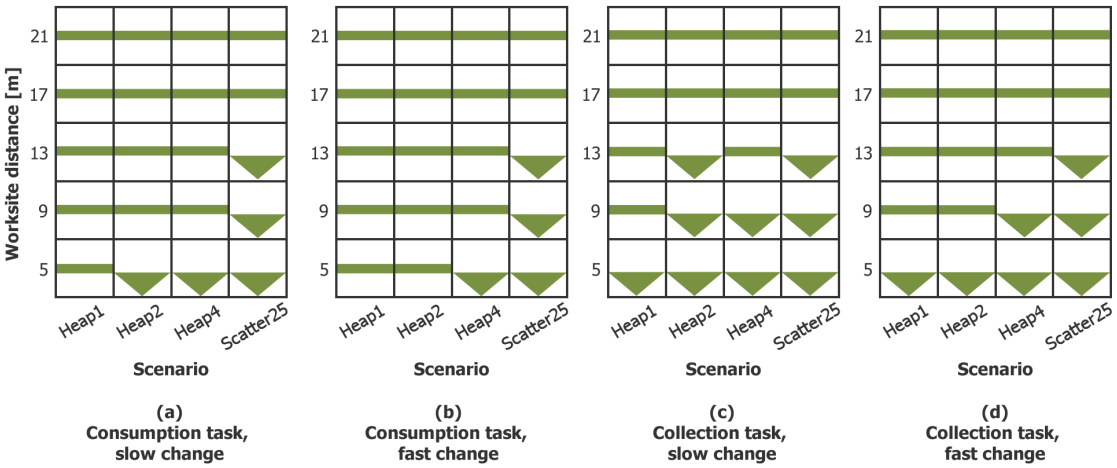


Figure F.19: The effect of Opportunism on the performance of 10-robot bee swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure F.27.

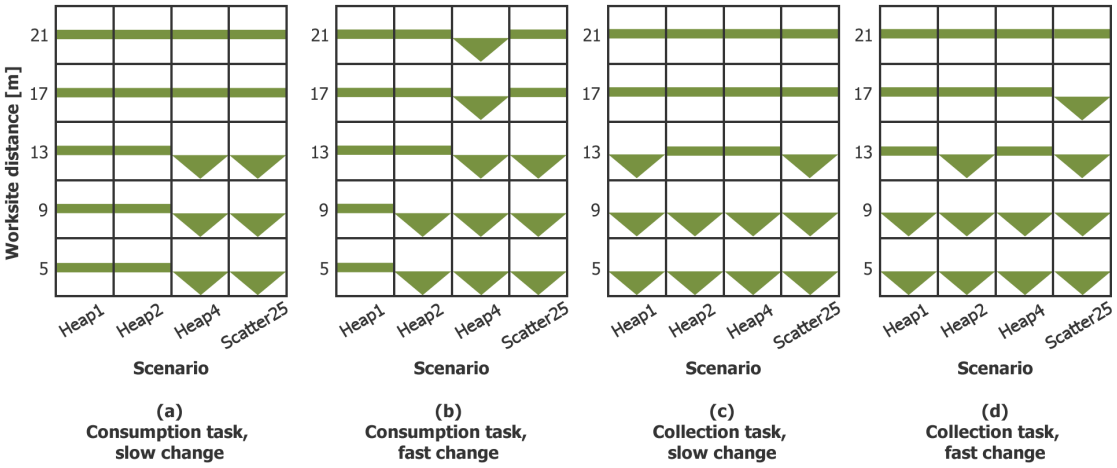


Figure F.20: The effect of Opportunism on the performance of 50-robot bee swarms. A downward-facing arrow indicates that Opportunism decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure F.29.

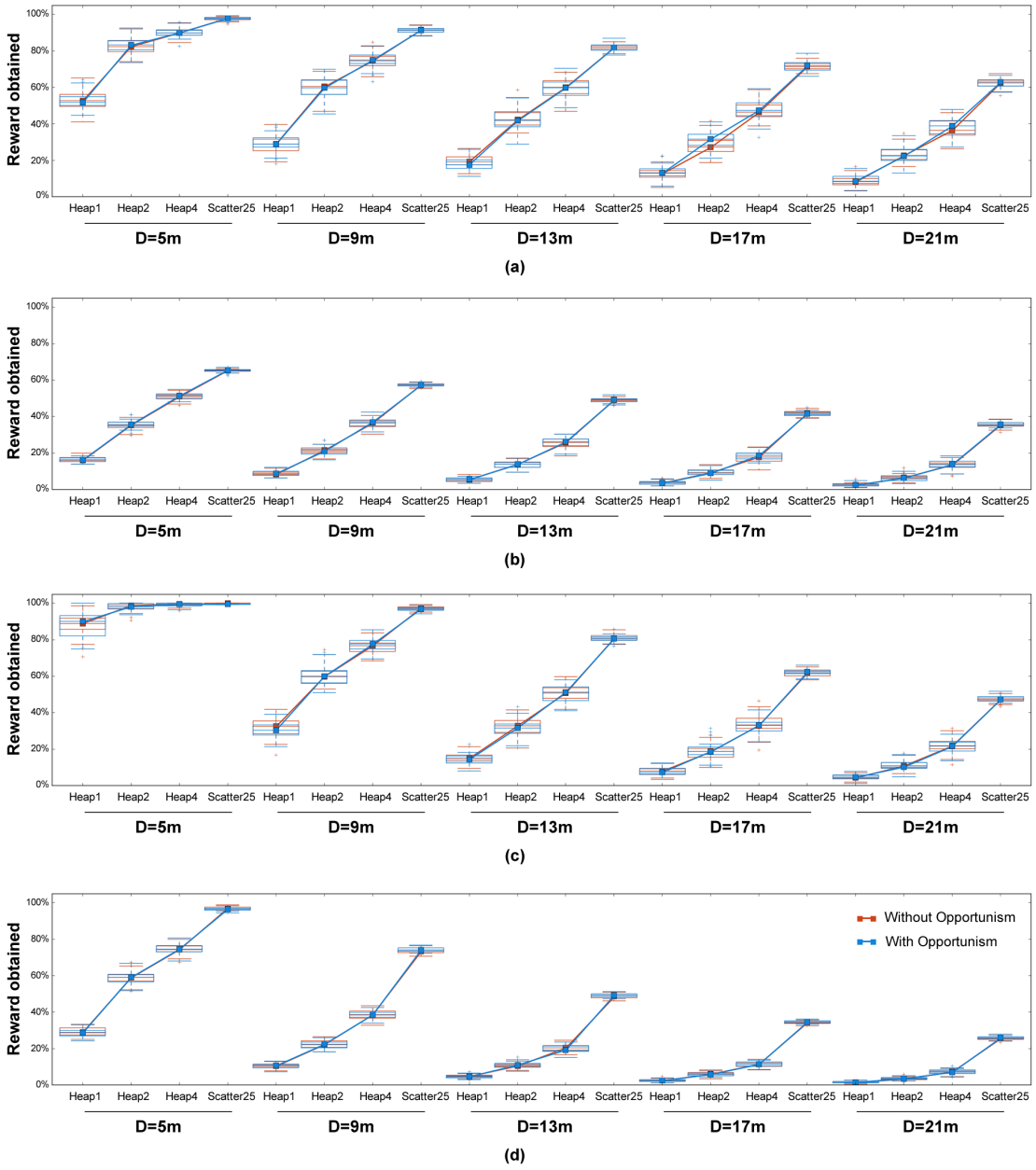


Figure F.21: The percentage of available reward obtained by 10-robot solitary swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of solitary swarms was generally unaffected by Opportunism.

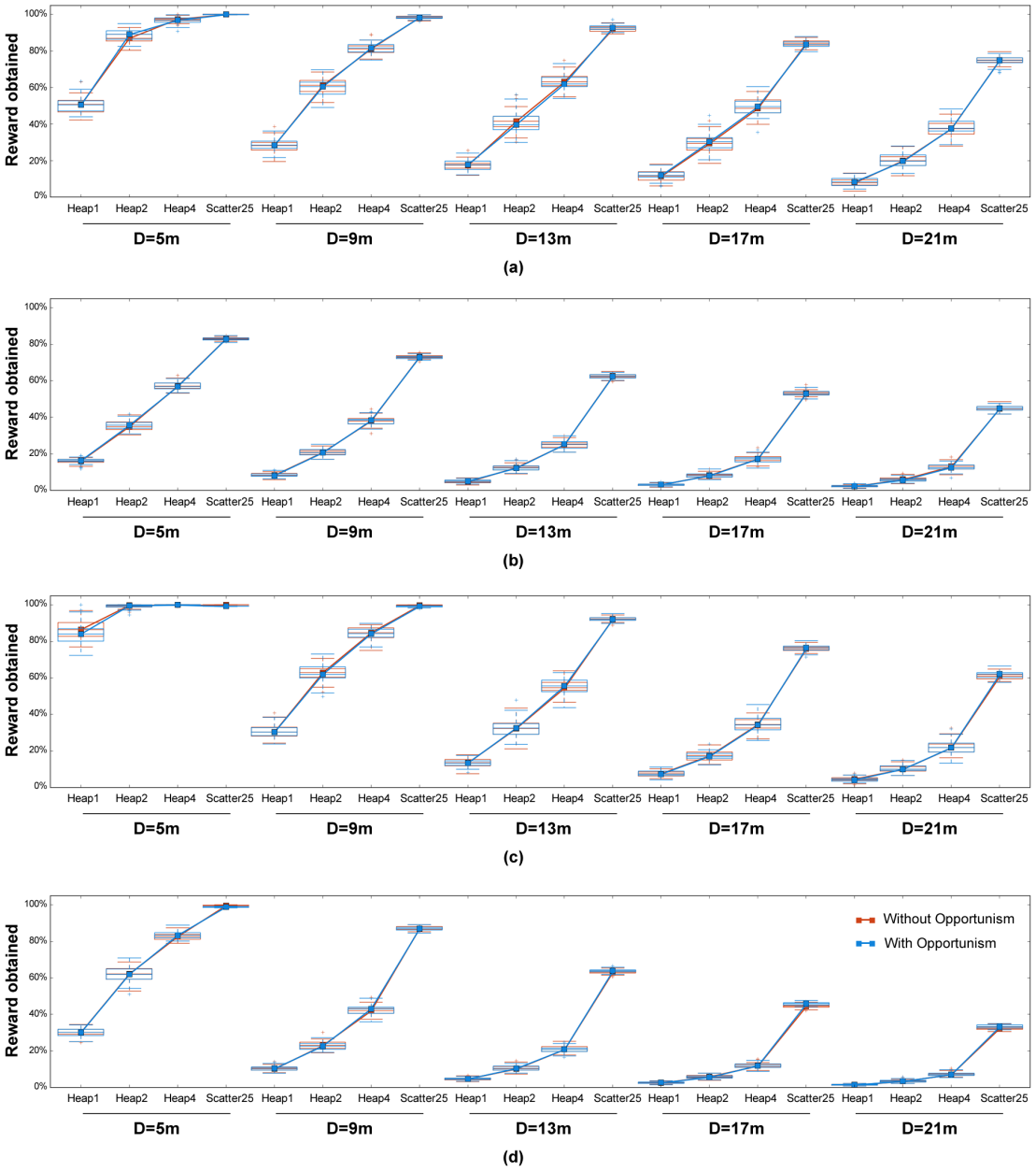


Figure F.22: The percentage of available reward obtained by 25-robot solitary swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of solitary swarms was generally unaffected by Opportunism.

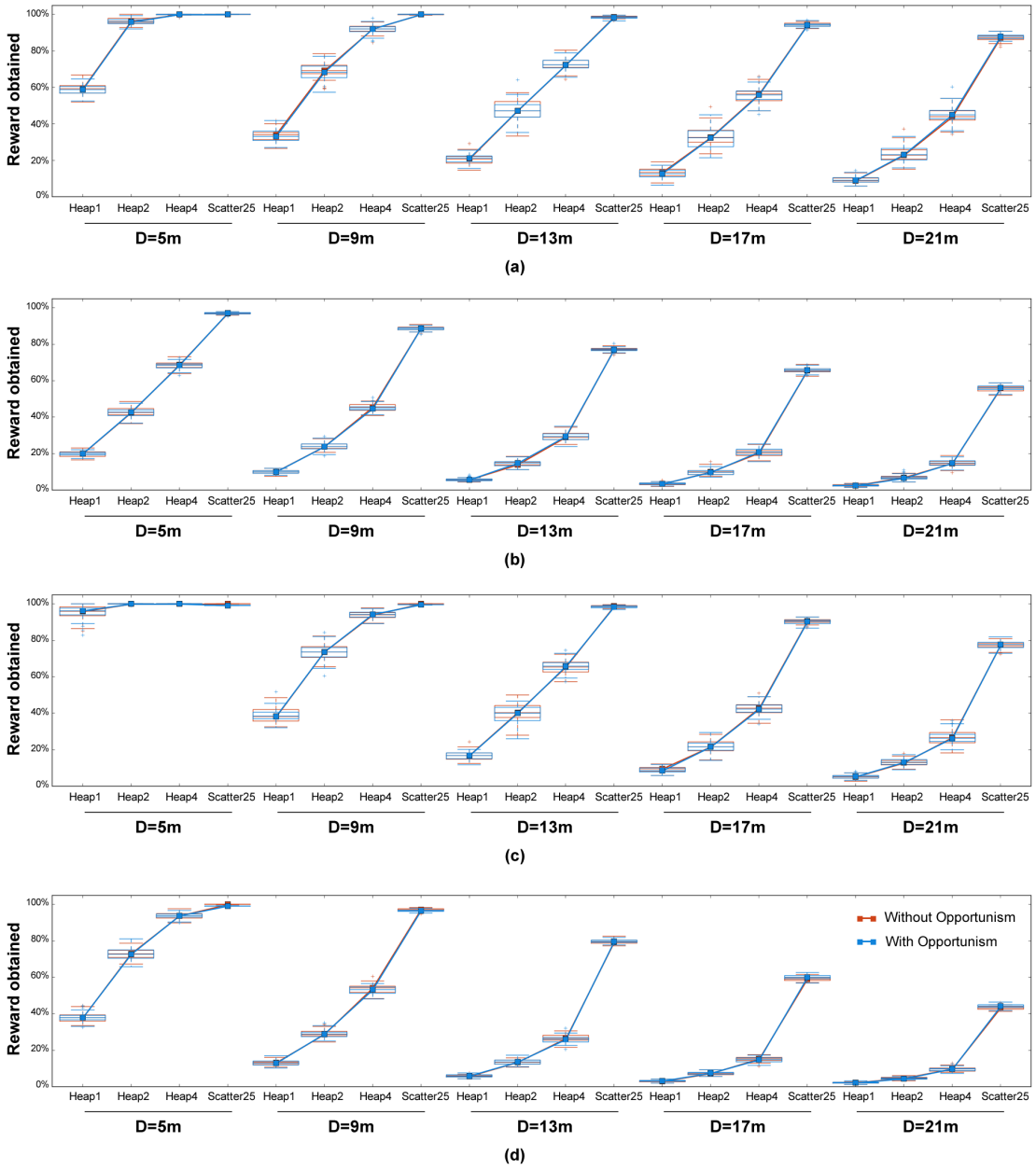


Figure F.23: The percentage of available reward obtained by 50-robot solitary swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of solitary swarms was generally unaffected by Opportunism.

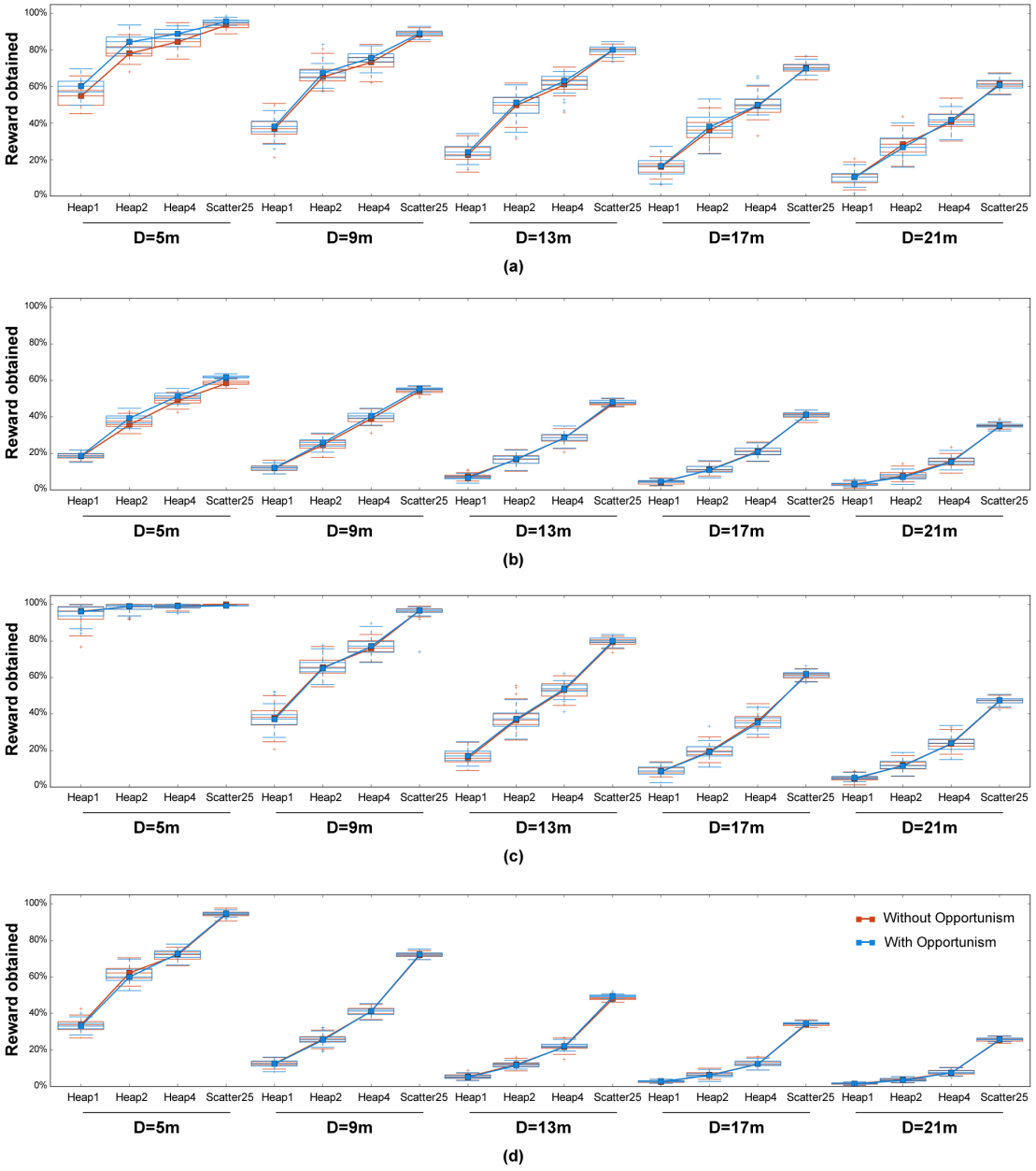


Figure F.24: The percentage of available reward obtained by 10-robot local broadcaster swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of local broadcasters was better with Opportunism in the consumption task when D was small.

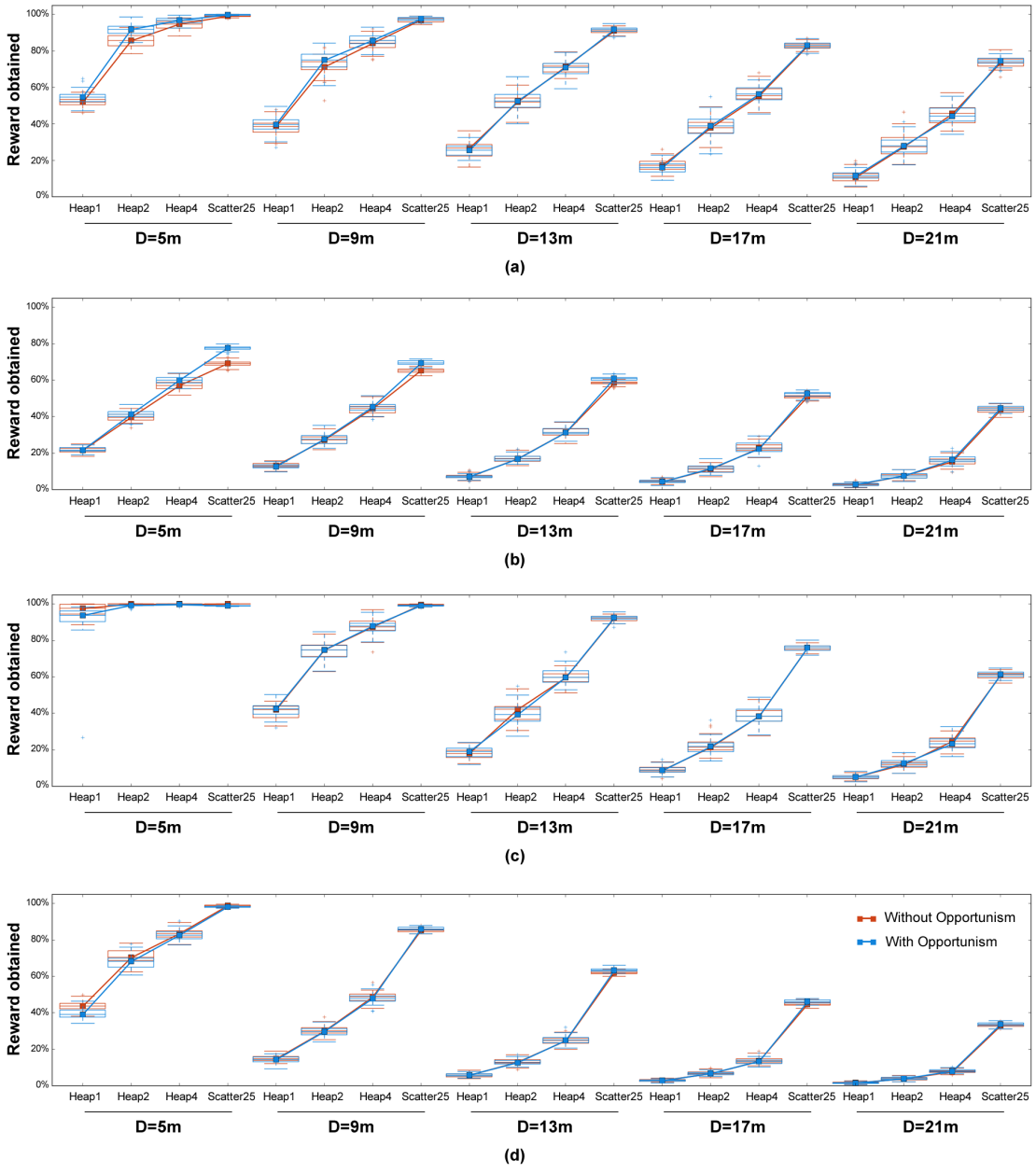


Figure F.25: The percentage of available reward obtained by 25-robot local broadcaster swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of local broadcasters was better with Opportunism in the consumption task in Scatter25 scenarios and in some Heap scenarios when D was small. The performance was worse with Opportunism, in the collection task in some Heap scenarios when D was small.

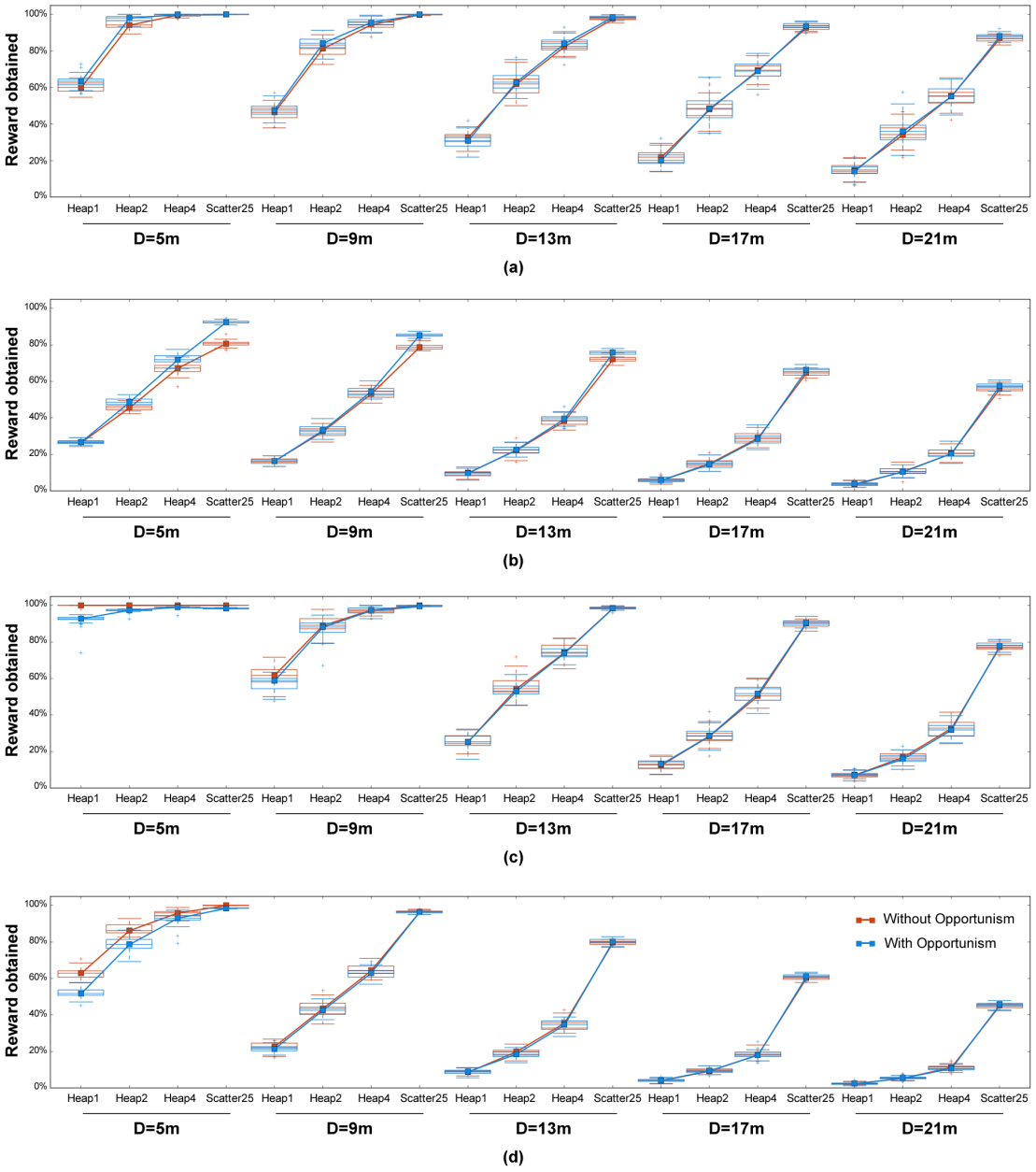


Figure F.26: The percentage of available reward obtained by 50-robot local broadcaster swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance was worse with Opportunism, in the collection task in some Heap scenarios when D was small.

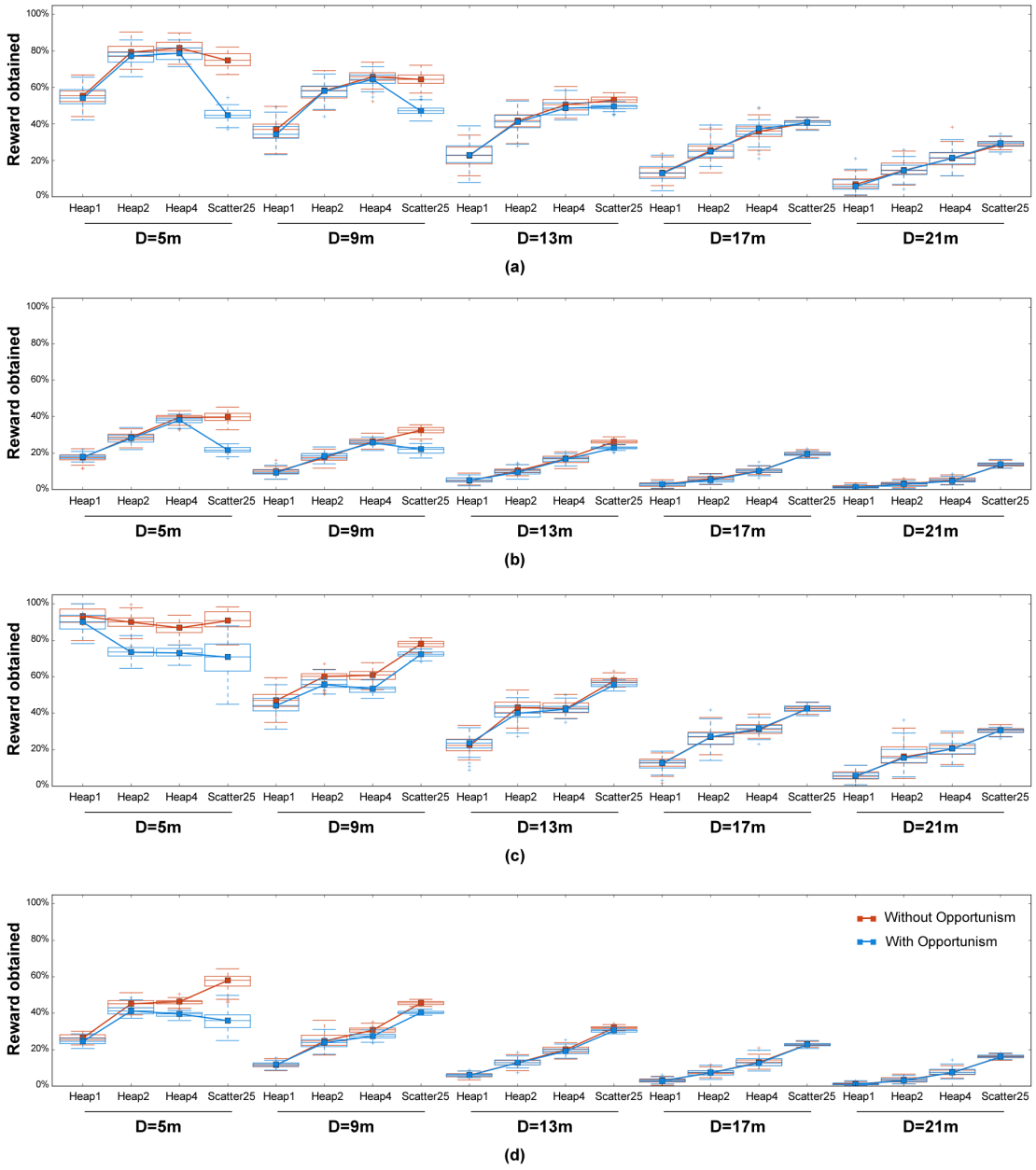


Figure F.27: The percentage of available reward obtained by 10-robot bee swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of bee swarms was worse with Opportunism in many scenarios, especially when D was small and when the robots performed the collection task.

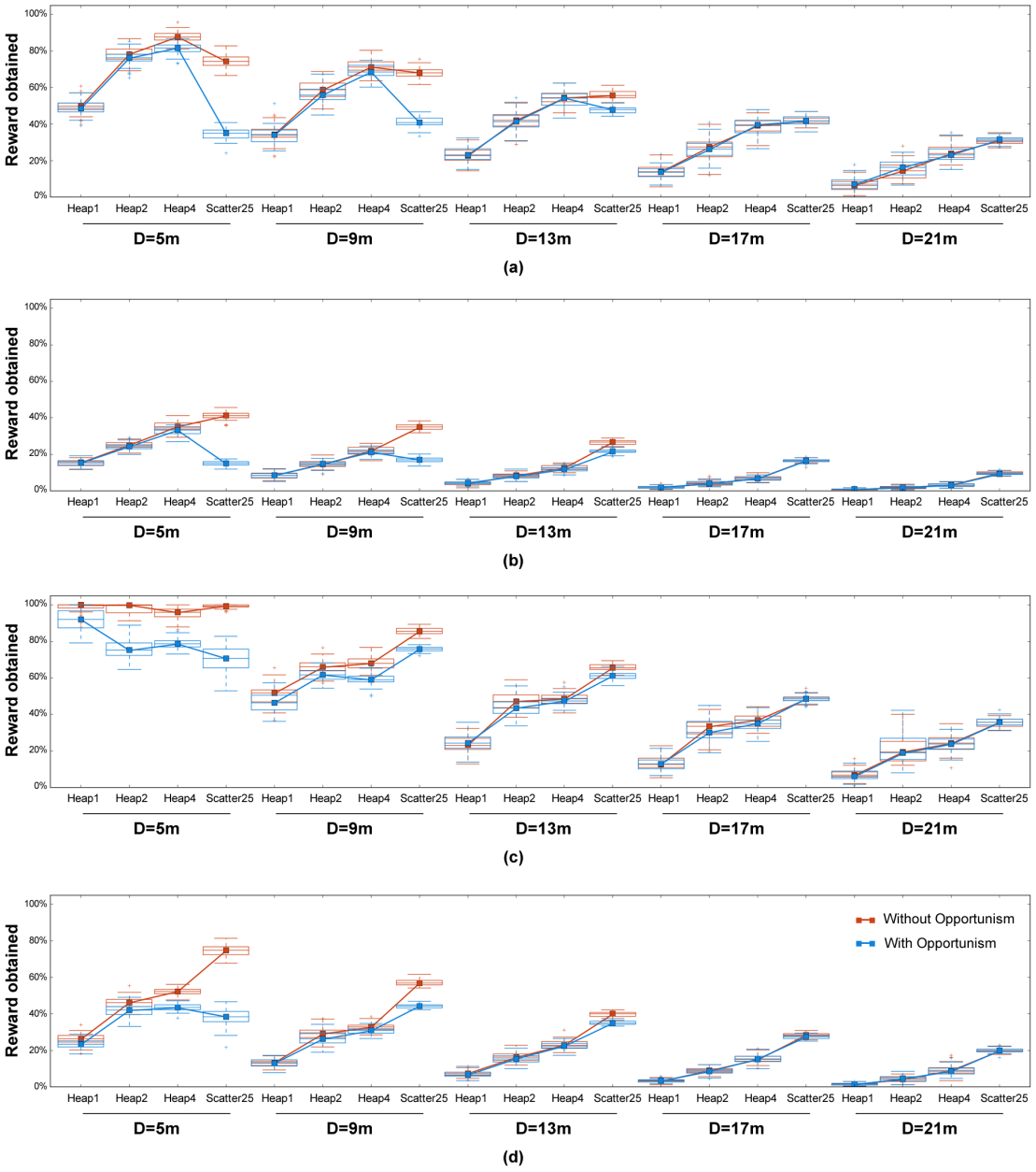


Figure F.28: The percentage of available reward obtained by 25-robot bee swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of bee swarms was worse with Opportunism in many scenarios, especially when D was small and when the robots performed the collection task.

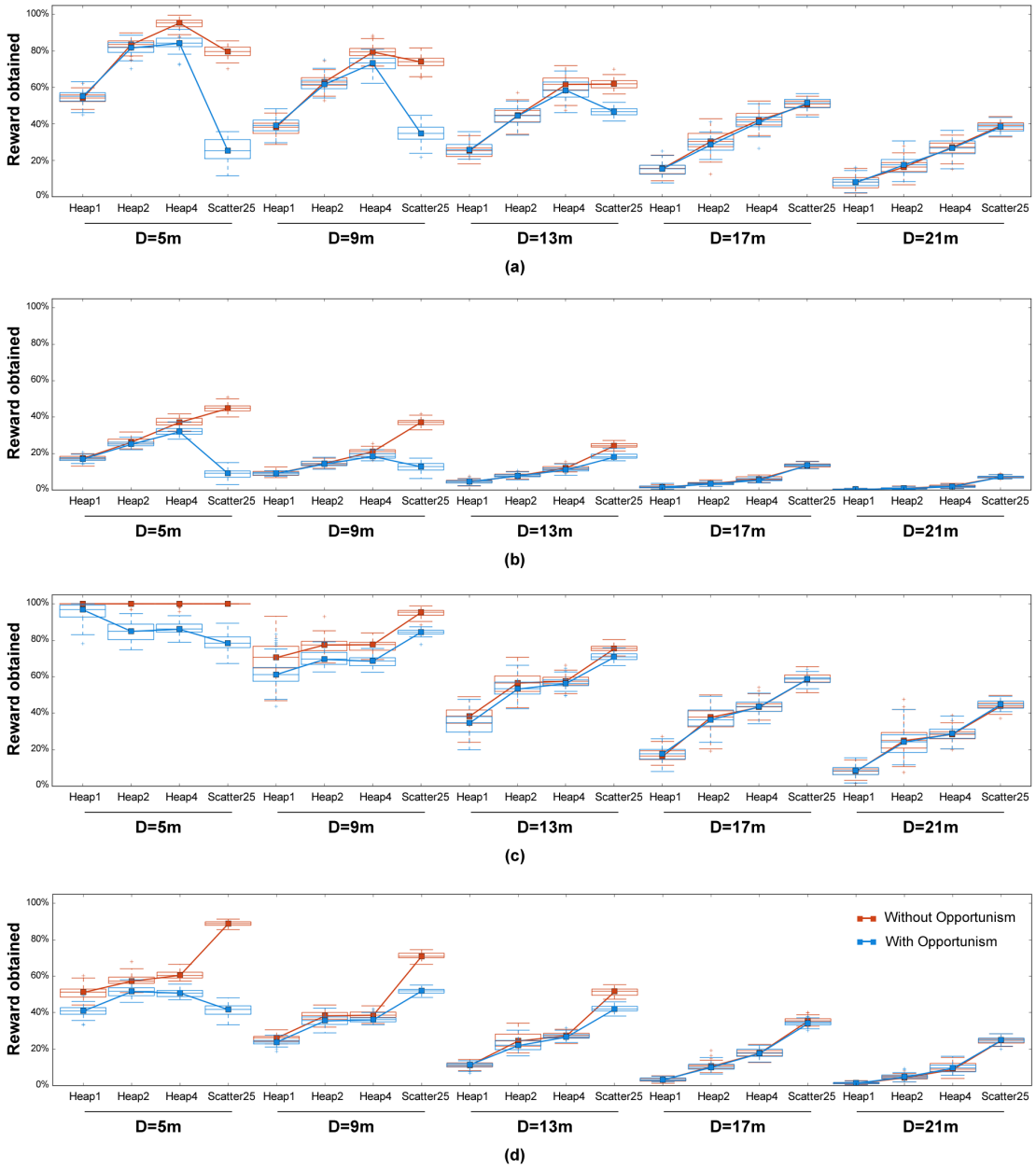


Figure F.29: The percentage of available reward obtained by 50-robot bee swarms with and without Opportunism in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of bee swarms was worse with Opportunism in many scenarios, especially when D was small and when the robots performed the collection task.

Appendix G

Supporting graphs for the analysis of the Anticipation add-on strategy

G.1 Information gain rate

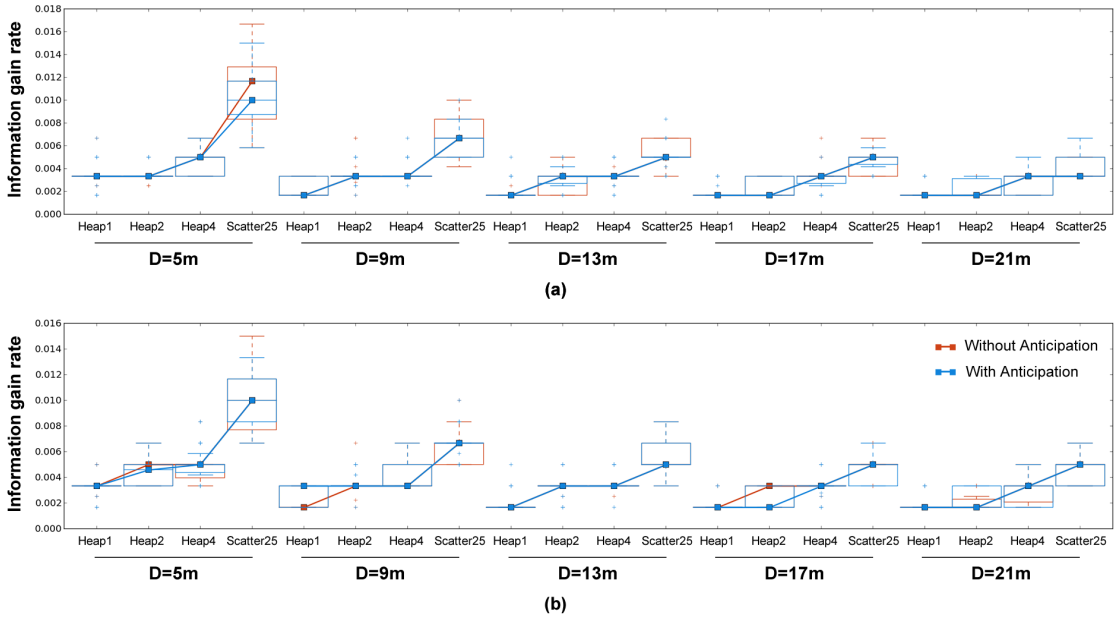


Figure G.1: Information gain rate, i, of 10-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The i of solitary swarms was not significantly affected by Anticipation.

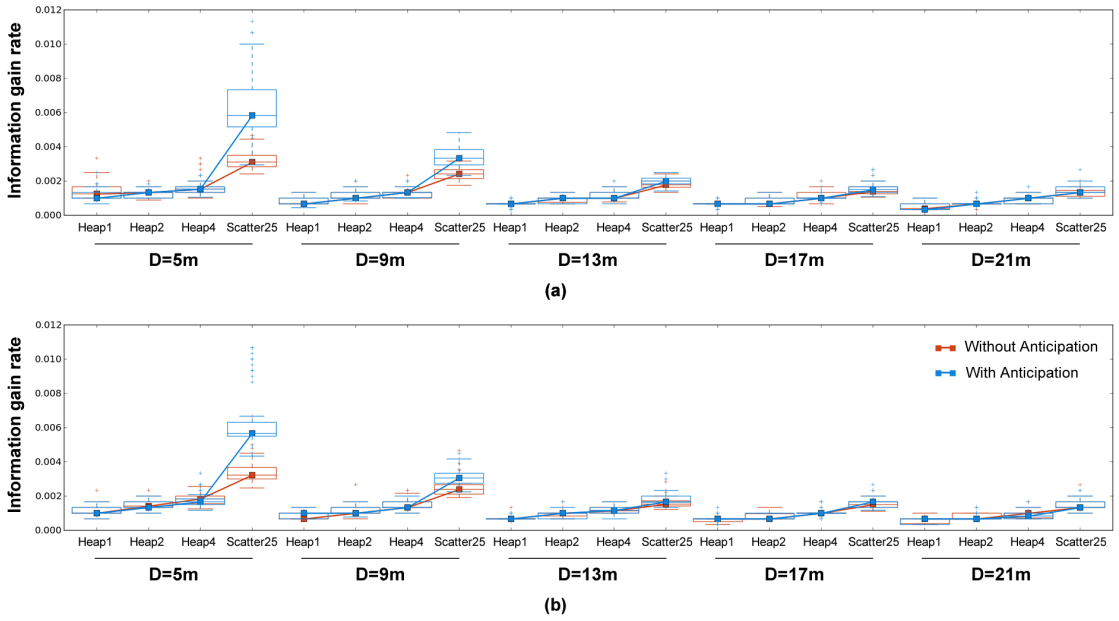


Figure G.2: Information gain rate, i, of 50-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The i of solitary swarms was higher with Anticipation in Scatter25 scenarios with a small D.

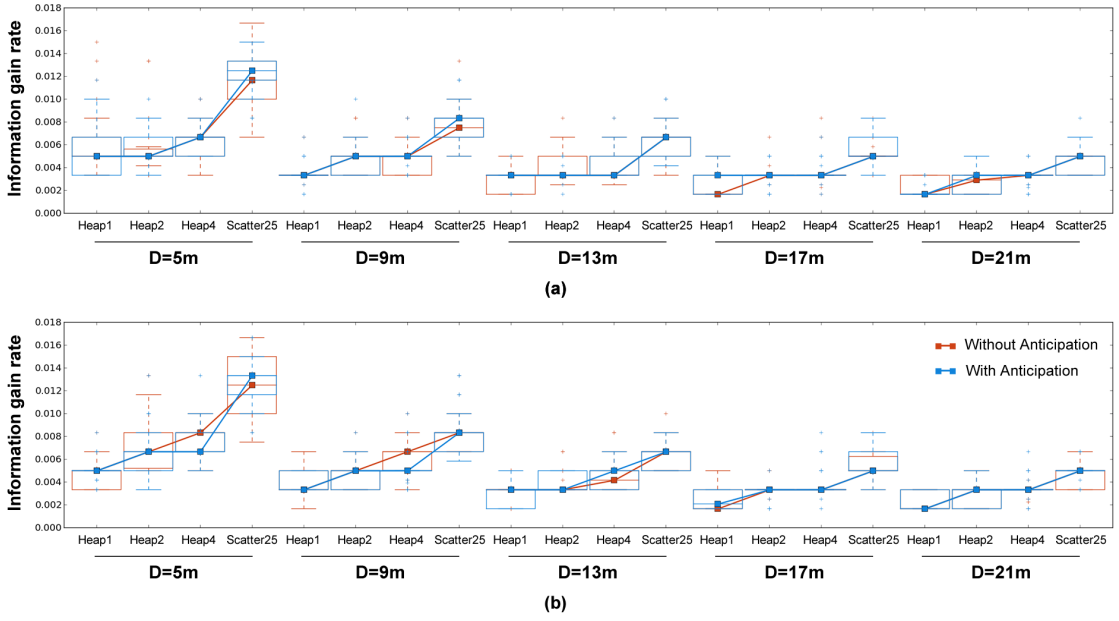


Figure G.3: Information gain rate, i, of 10-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The i of local broadcasters was not significantly affected by Anticipation.

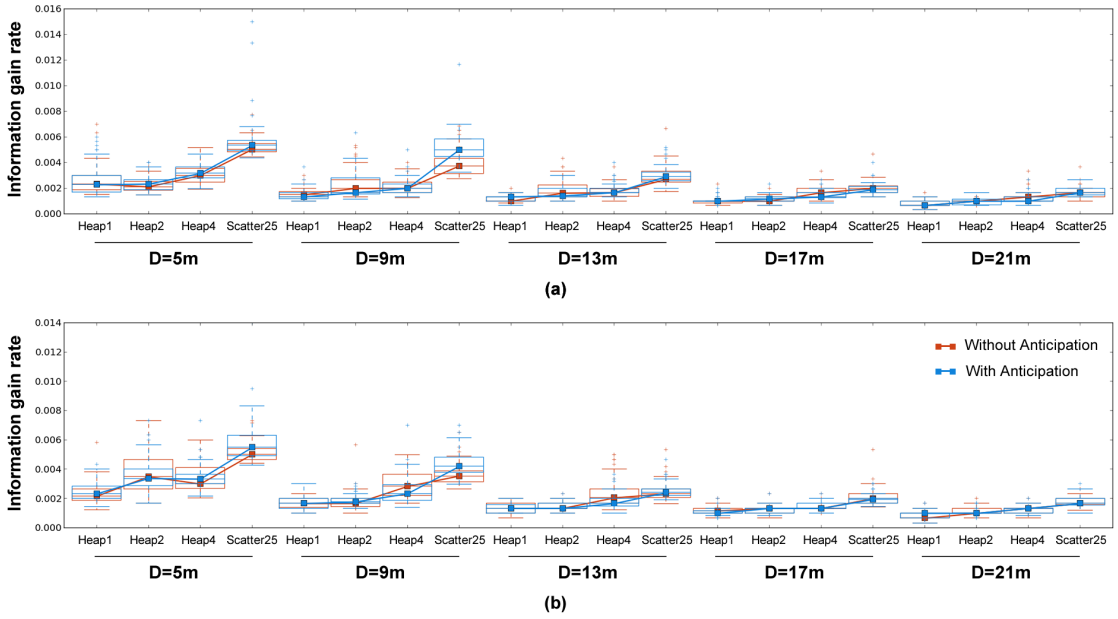


Figure G.4: Information gain rate, i, of 50-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The i of local broadcasters was higher with Anticipation in Scatter25 scenarios when $D = 9$m.

G.2 Misplacement cost coefficient

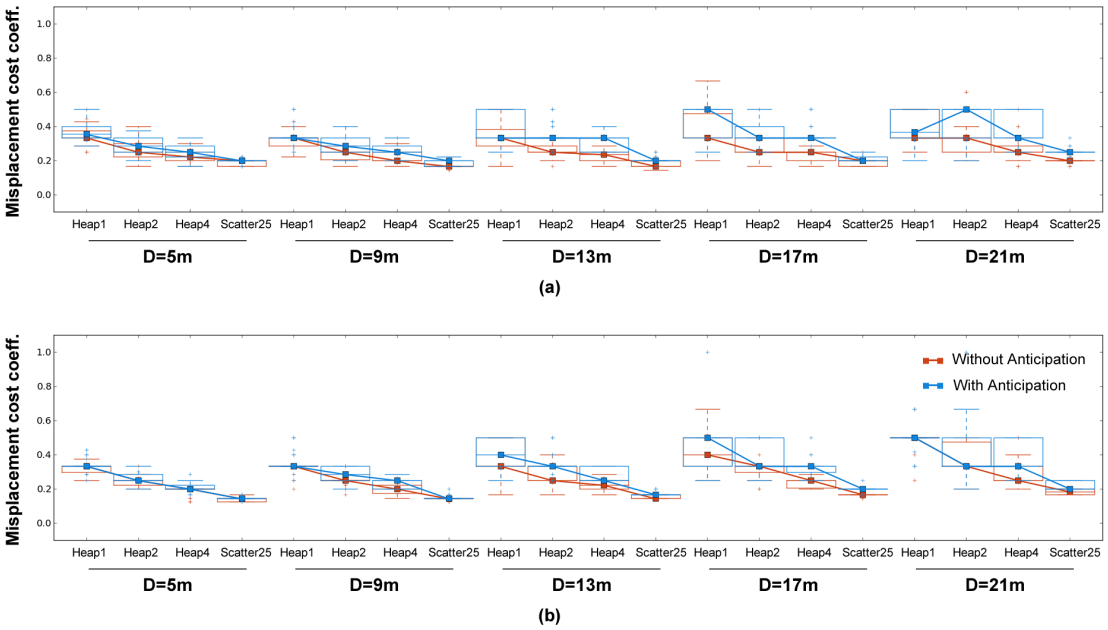


Figure G.5: Misplacement cost coefficient, m, of 10-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The m of local broadcasters was higher with Anticipation in Heap2, Heap4 and Scatter25 scenarios when $D \geq 13$m and when the environment changed slowly.

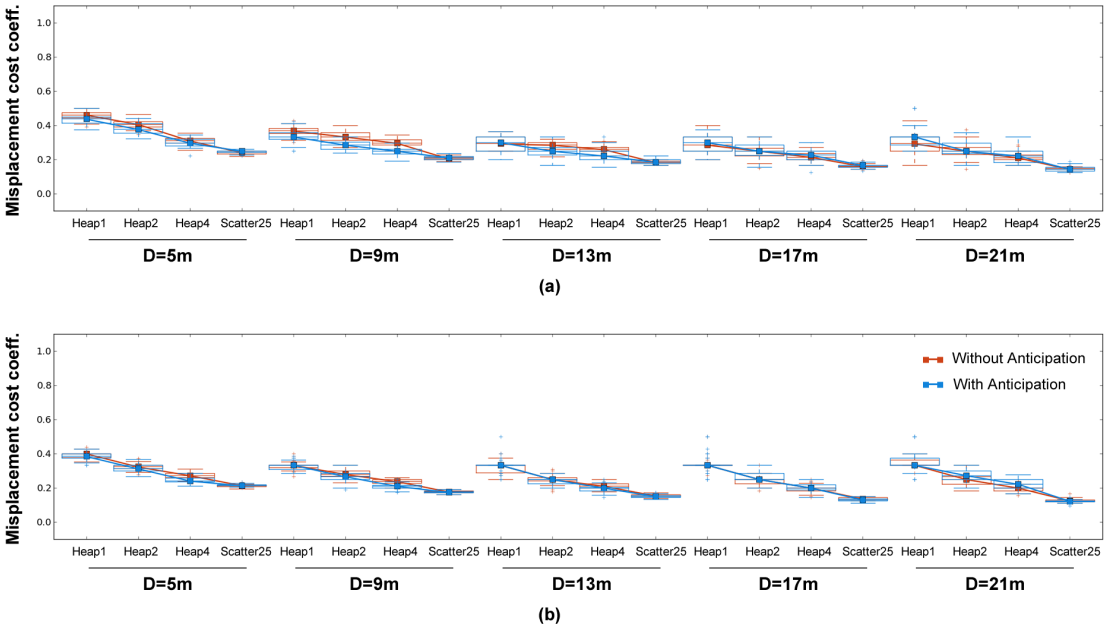


Figure G.6: Misplacement cost coefficient, m, of 50-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The m of local broadcasters was lower with Anticipation in some Heap scenarios when $D \leq 13$m and when the environment changed slowly.

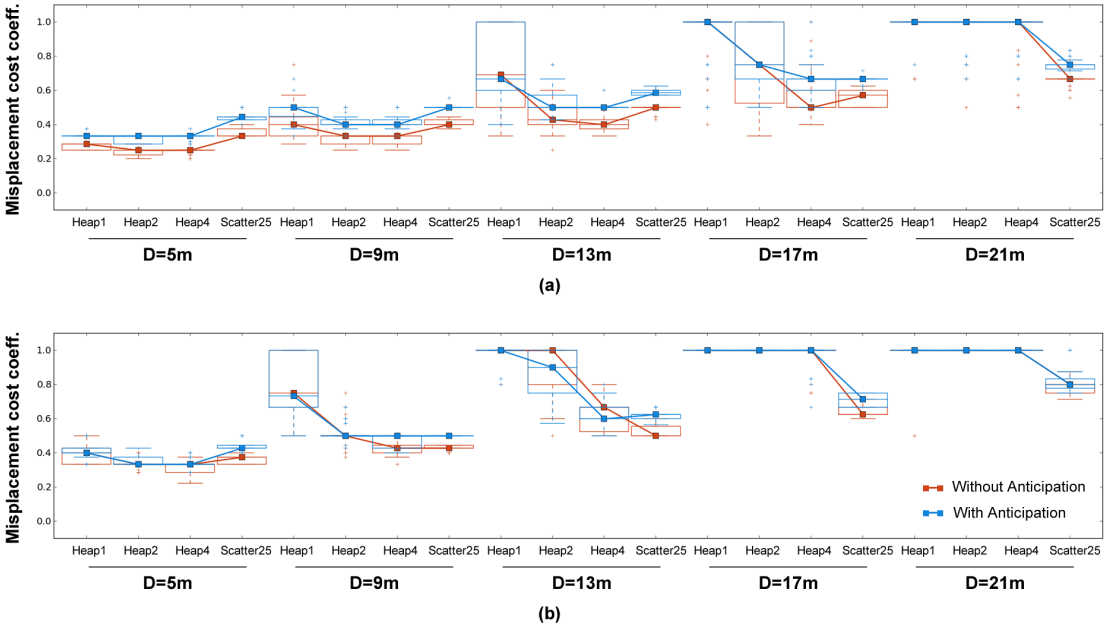


Figure G.7: Misplacement cost coefficient, m, of 10-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was higher with Anticipation in most scenarios, especially when worksites were close to the base and when the environment changed slowly.

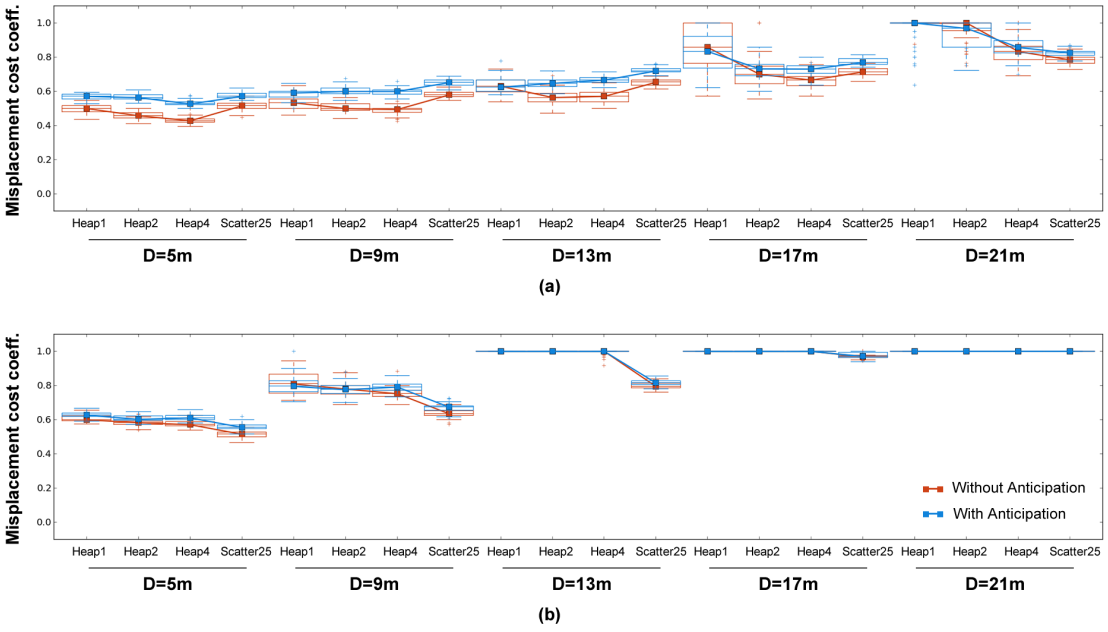


Figure G.8: Misplacement cost coefficient, m, of 50-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The m of bee swarms was higher with Anticipation in most scenarios, especially when worksites were close to the base and when the environment changed slowly.

G.3 Opportunity cost

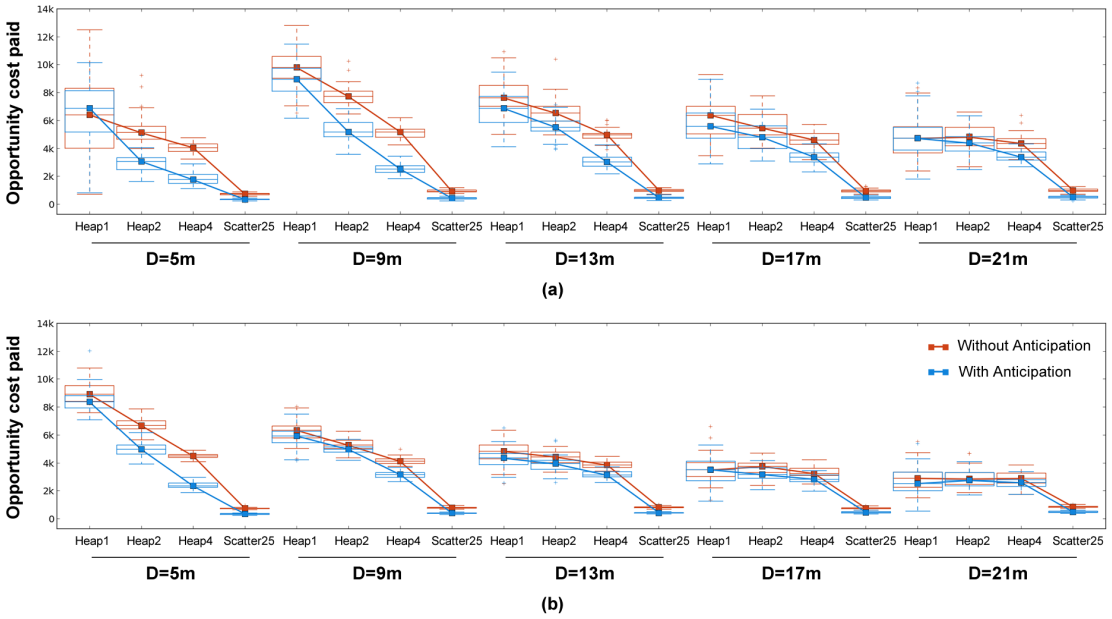


Figure G.9: Opportunity cost, C_O, of 10-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by solitary swarms was smaller with Anticipation in most of the scenarios, especially when the environment changed slowly.

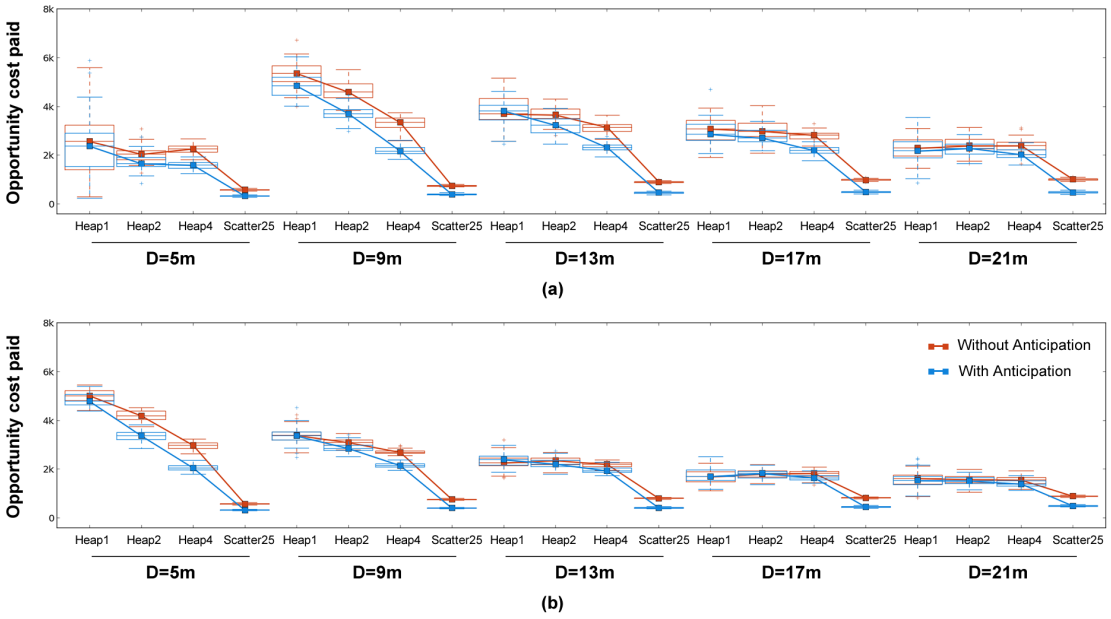


Figure G.10: Opportunity cost, C_O, of 50-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by solitary swarms was smaller with Anticipation in most of the scenarios, especially when the environment changed slowly.

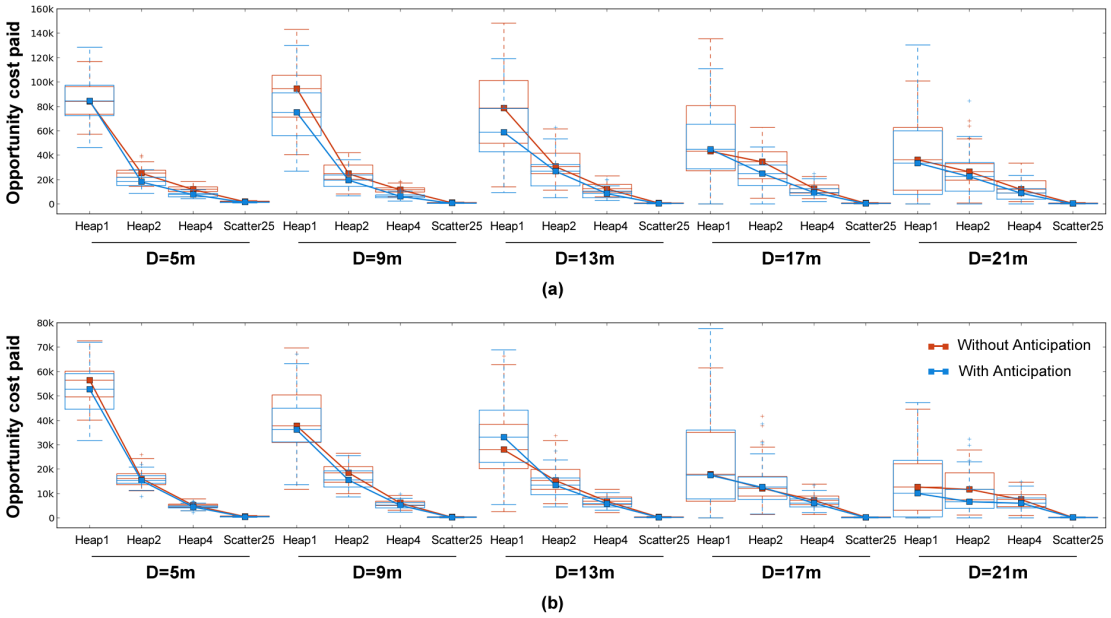


Figure G.11: Opportunity cost, C_O, of 10-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by local broadcasters was smaller with Anticipation in most Heap scenarios when $D \leq 9$m and when the environment changed slowly.

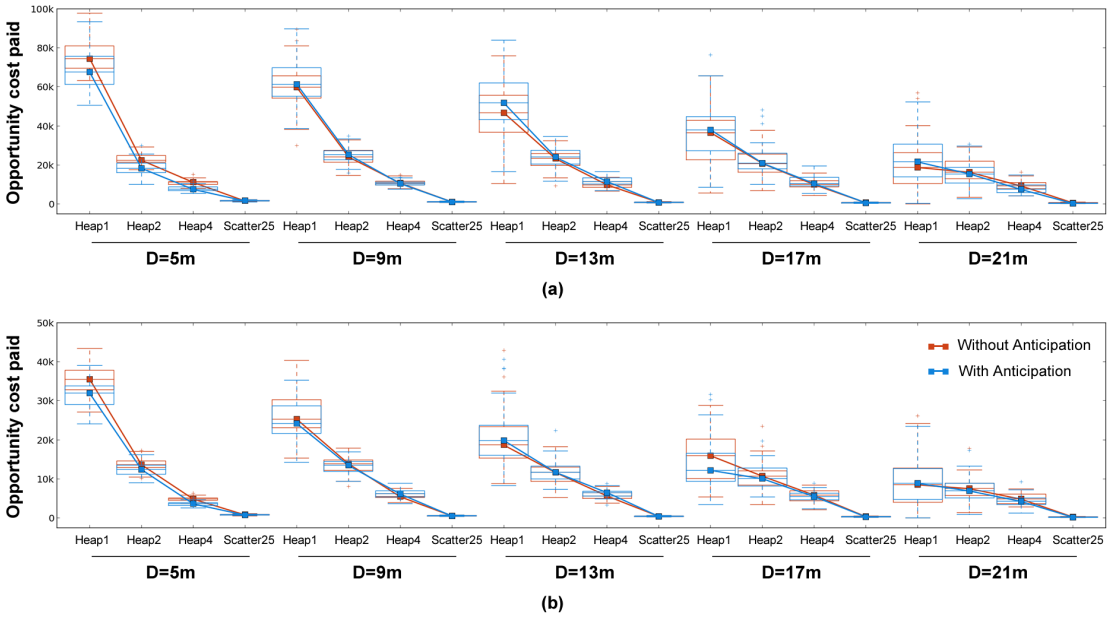


Figure G.12: Opportunity cost, C_O, of 25-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by local broadcasters was smaller with Anticipation in most Heap scenarios when $D = 5$m.

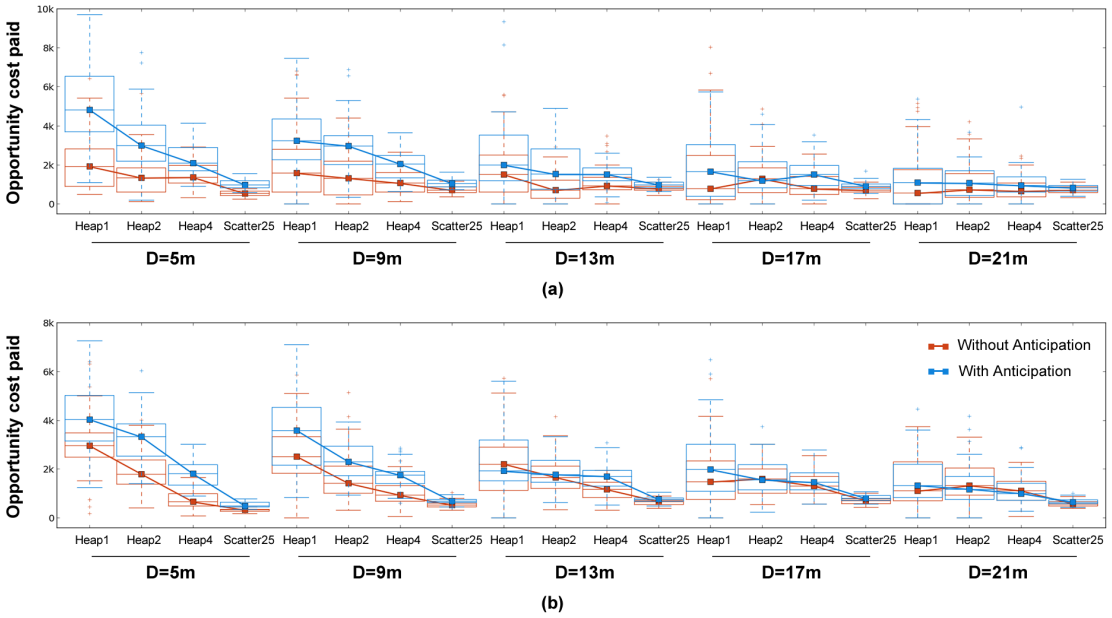


Figure G.13: Opportunity cost, C_O, of 10-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by bee swarms was larger with Anticipation in most scenarios, especially when D was small or when the environment changed slowly.

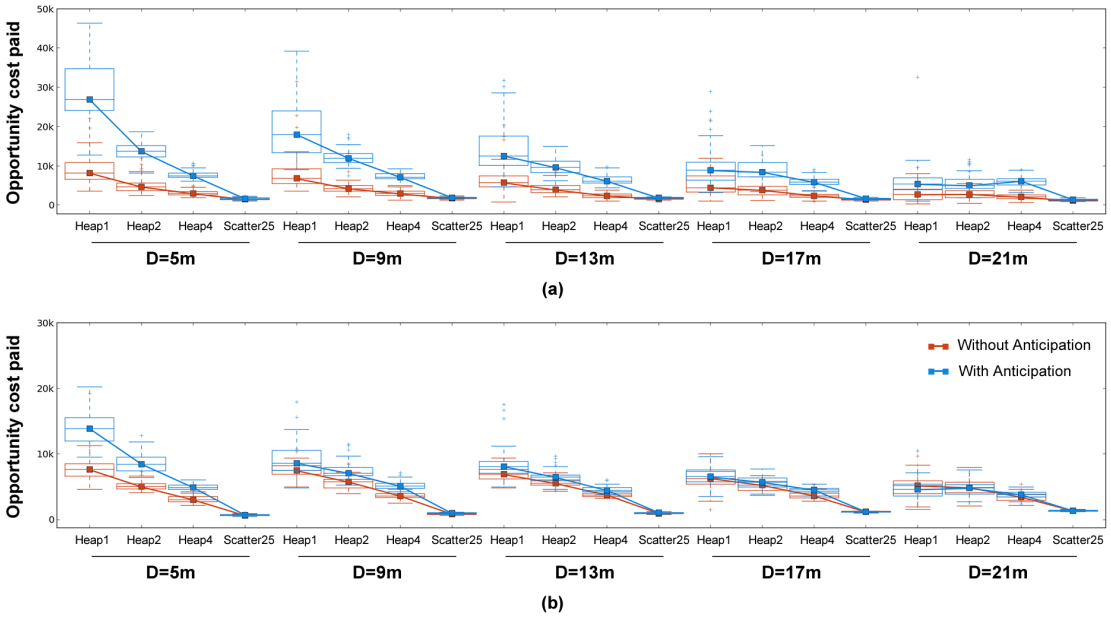


Figure G.14: Opportunity cost, C_O, of 50-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_O paid by bee swarms was larger with Anticipation in most scenarios, especially when D was small or when the environment changed slowly.

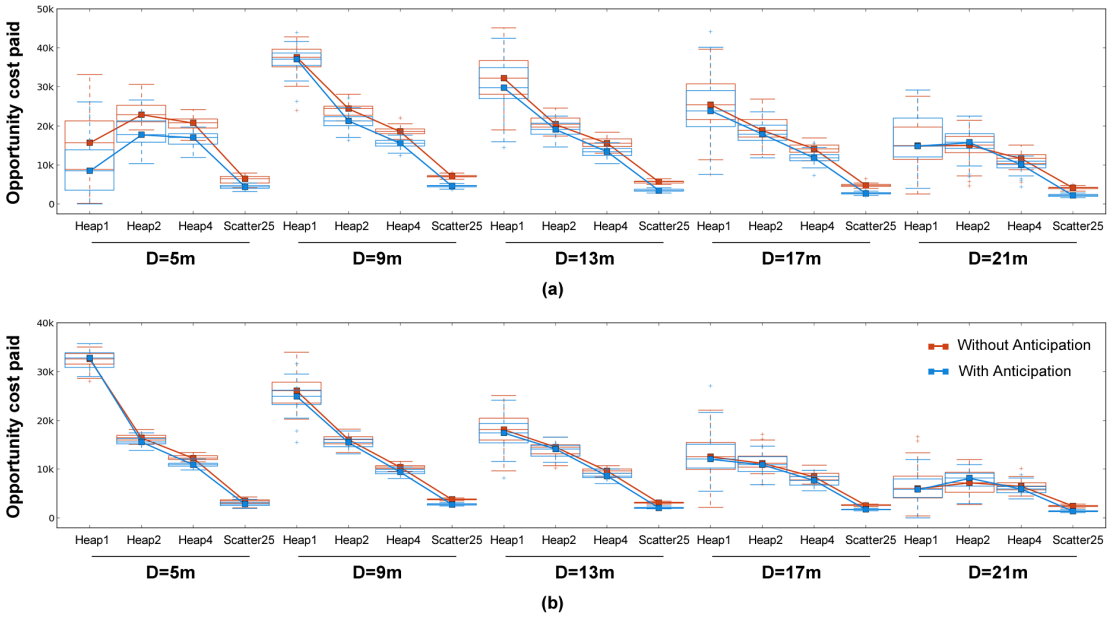


Figure G.15: Opportunity cost, C_O, of 10-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by bee swarms was smaller with Anticipation in some scenarios, mostly when D was small and when the environment changed slowly. The C_O paid was unaffected by Anticipation in other scenarios.

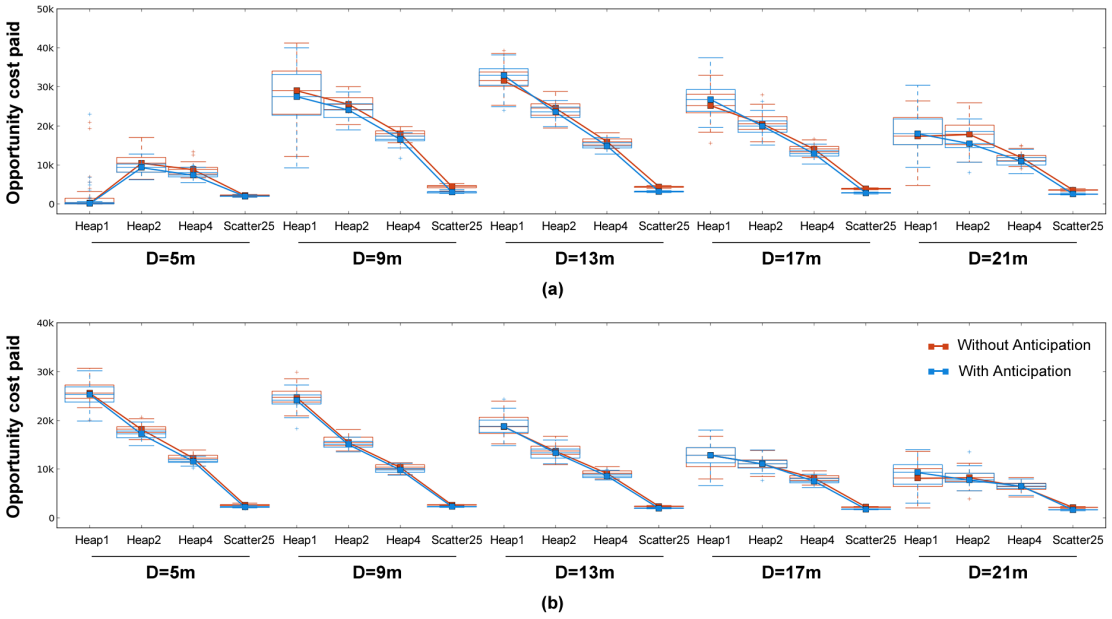


Figure G.16: Opportunity cost, C_O, of 50-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_O paid by bee swarms was smaller with Anticipation in some Heap4 and Scatter25 scenarios. The C_O paid was unaffected by Anticipation in other scenarios.

G.4 Reward obtained

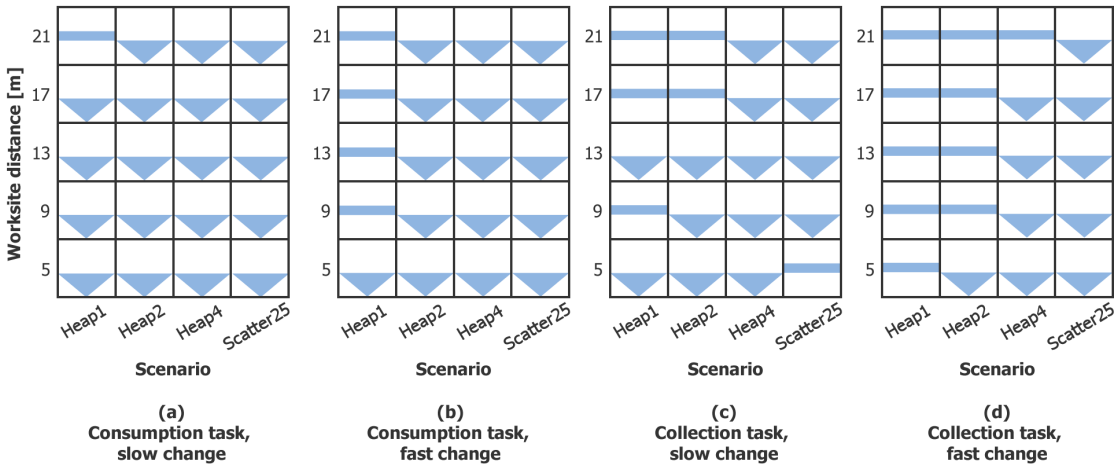


Figure G.17: The effect of Anticipation on the performance of 10-robot solitary swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure G.23.

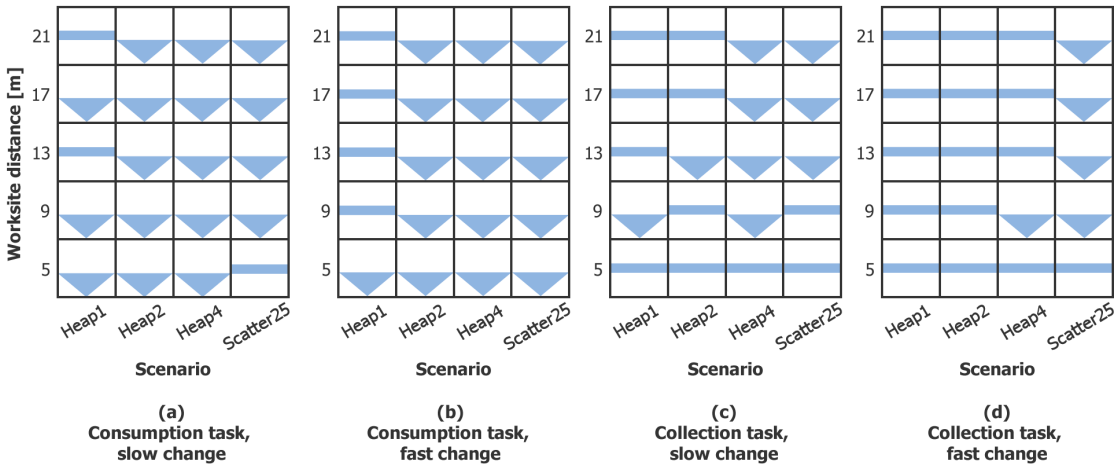


Figure G.18: The effect of Anticipation on the performance of 50-robot solitary swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure G.25.

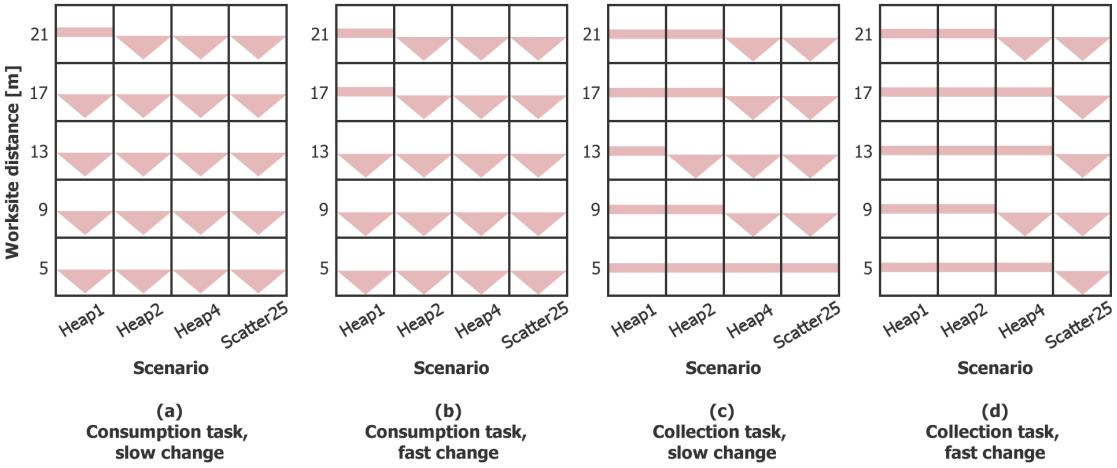


Figure G.19: The effect of Anticipation on the performance of 10-robot local broadcaster swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure G.26.

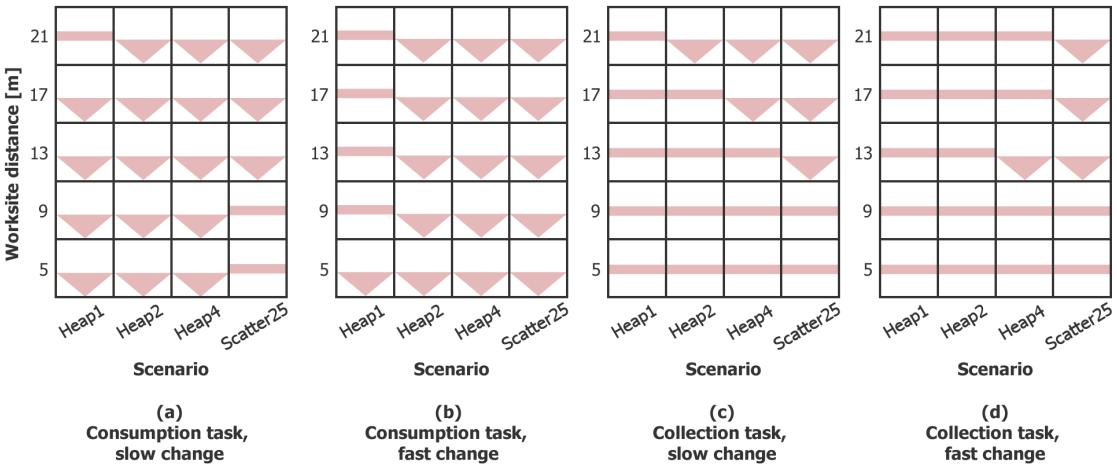


Figure G.20: The effect of Anticipation on the performance of 50-robot local broadcaster swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure G.28.

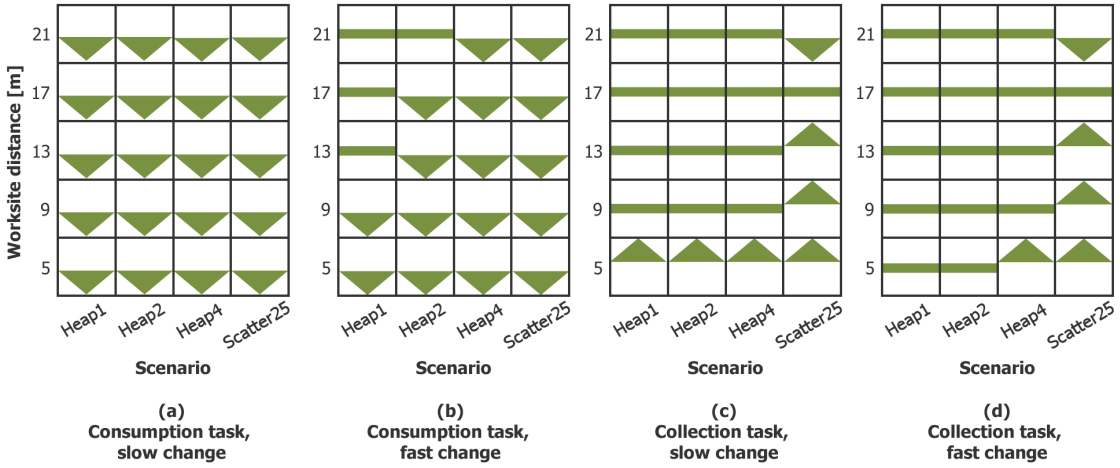


Figure G.21: The effect of Anticipation on the performance of 10-robot bee swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure G.29.

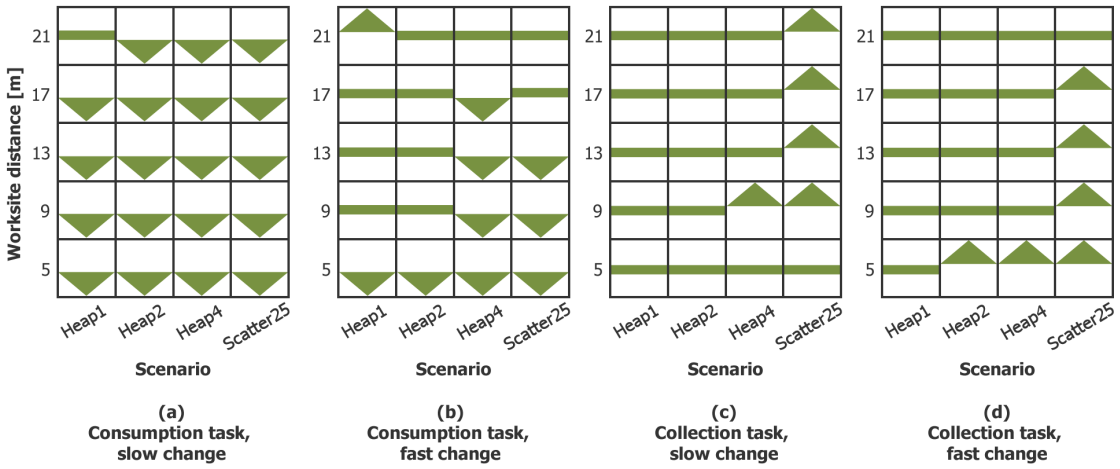


Figure G.22: The effect of Anticipation on the performance of 50-robot bee swarms. A downward-facing arrow indicates that Anticipation decreased the swarm performance, upward-facing arrow symbolises an improved performance and a horizontal bar indicates no significant effect on performance. Box plots of these results, that show the spread of the data over 50 experimental runs for each environment, are shown in Figure G.31.

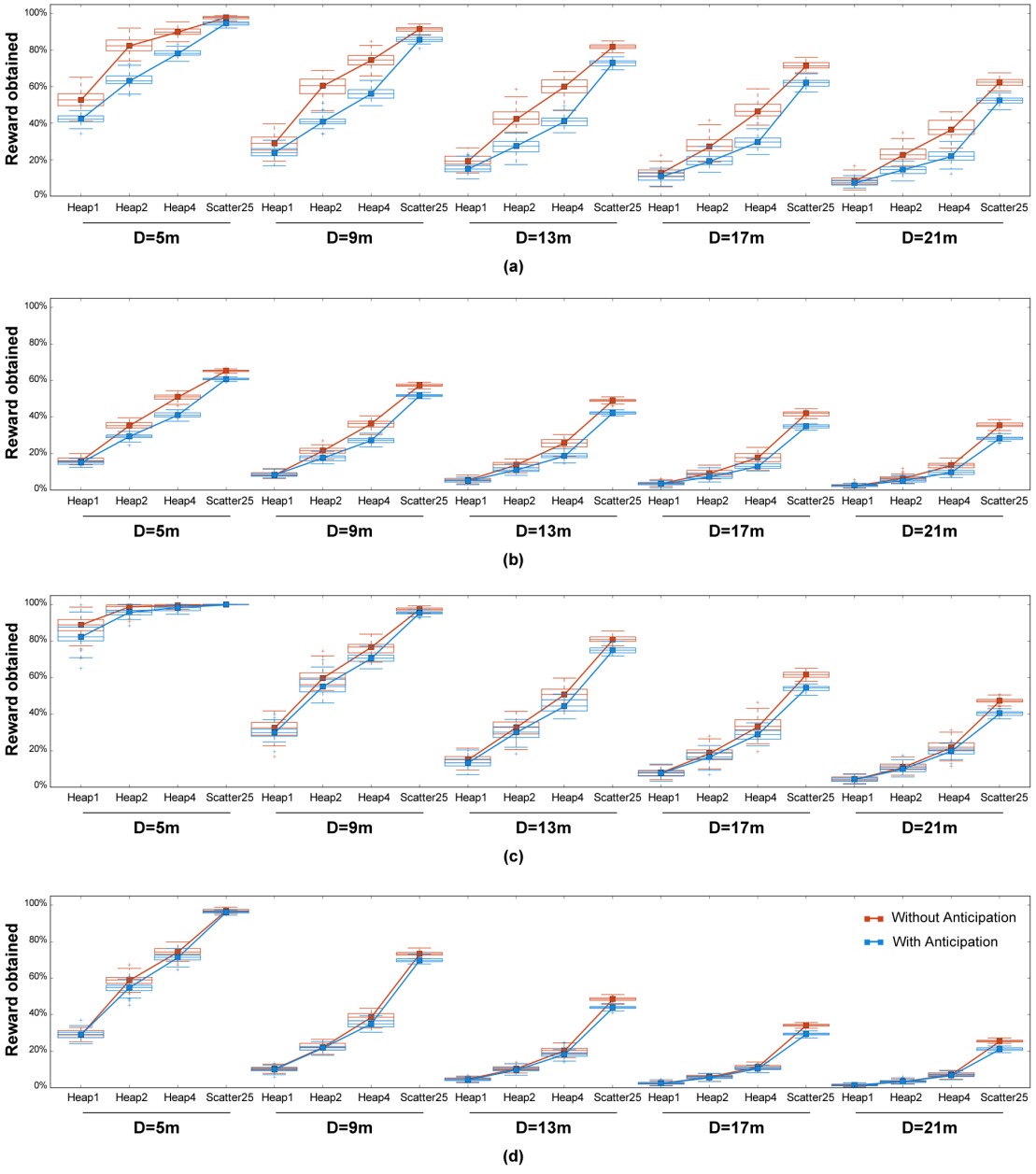


Figure G.23: The percentage of available reward obtained by 10-robot solitary swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of solitary swarms was worse with Anticipation in most scenarios, especially in the slow consumption task.

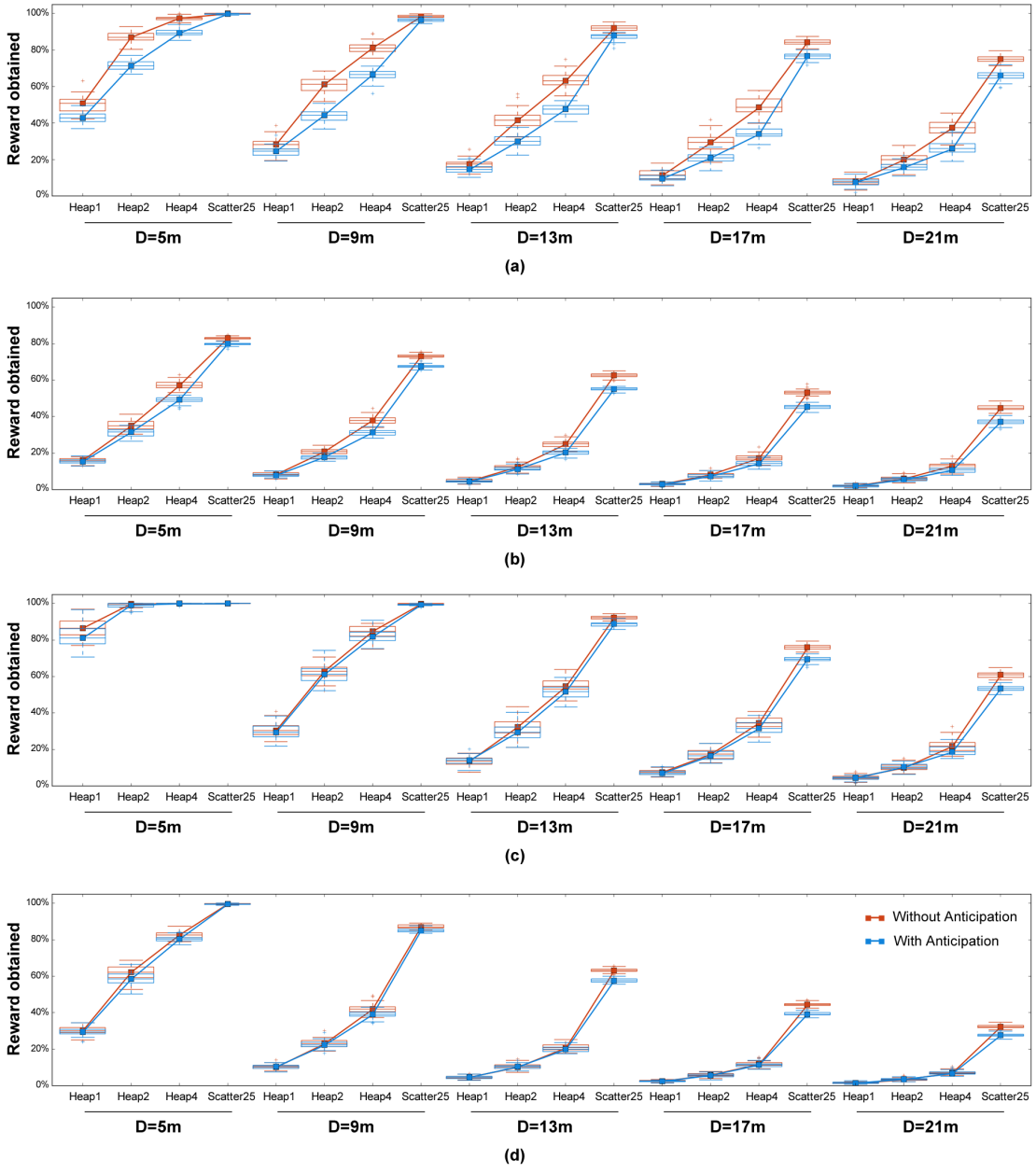


Figure G.24: The percentage of available reward obtained by 25-robot solitary swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of solitary swarms was worse with Anticipation in some scenarios, especially in the slow consumption task.

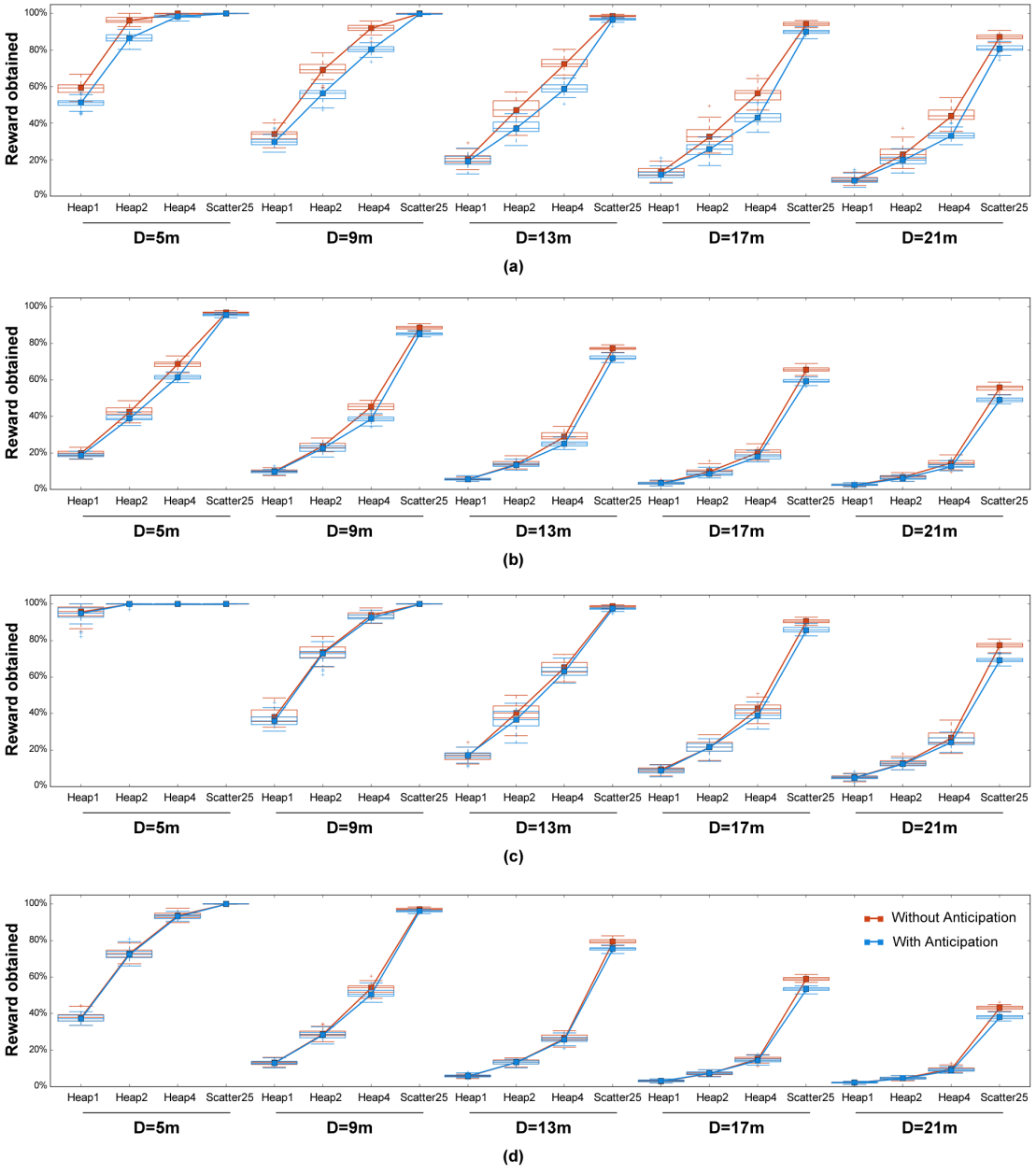


Figure G.25: The percentage of available reward obtained by 50-robot solitary swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of solitary swarms was worse with Anticipation in some scenarios, especially in the slow consumption task.

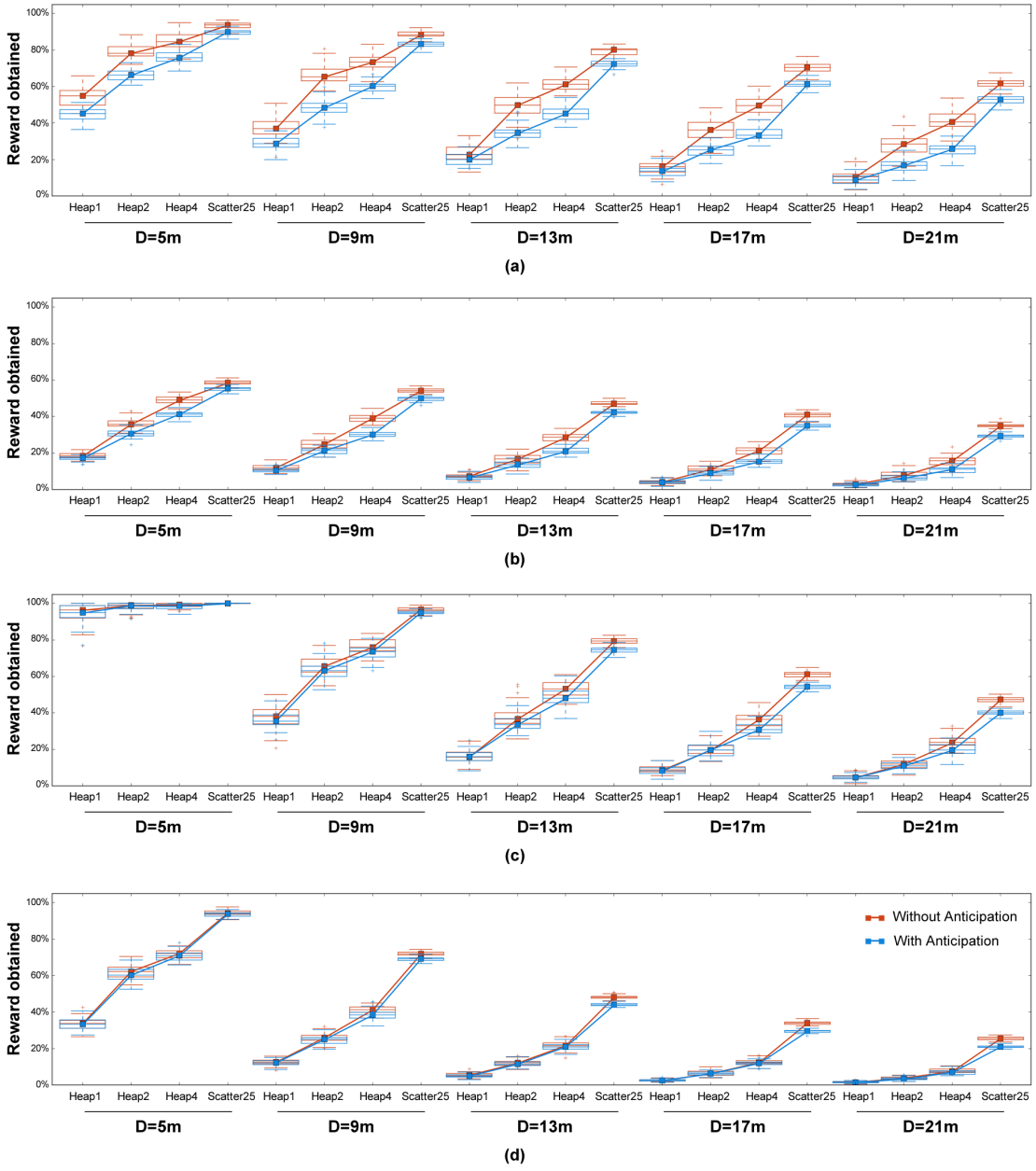


Figure G.26: The percentage of available reward obtained by 10-robot local broadcaster swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of local broadcasters was worse with Anticipation in some scenarios, especially in the slow consumption task.

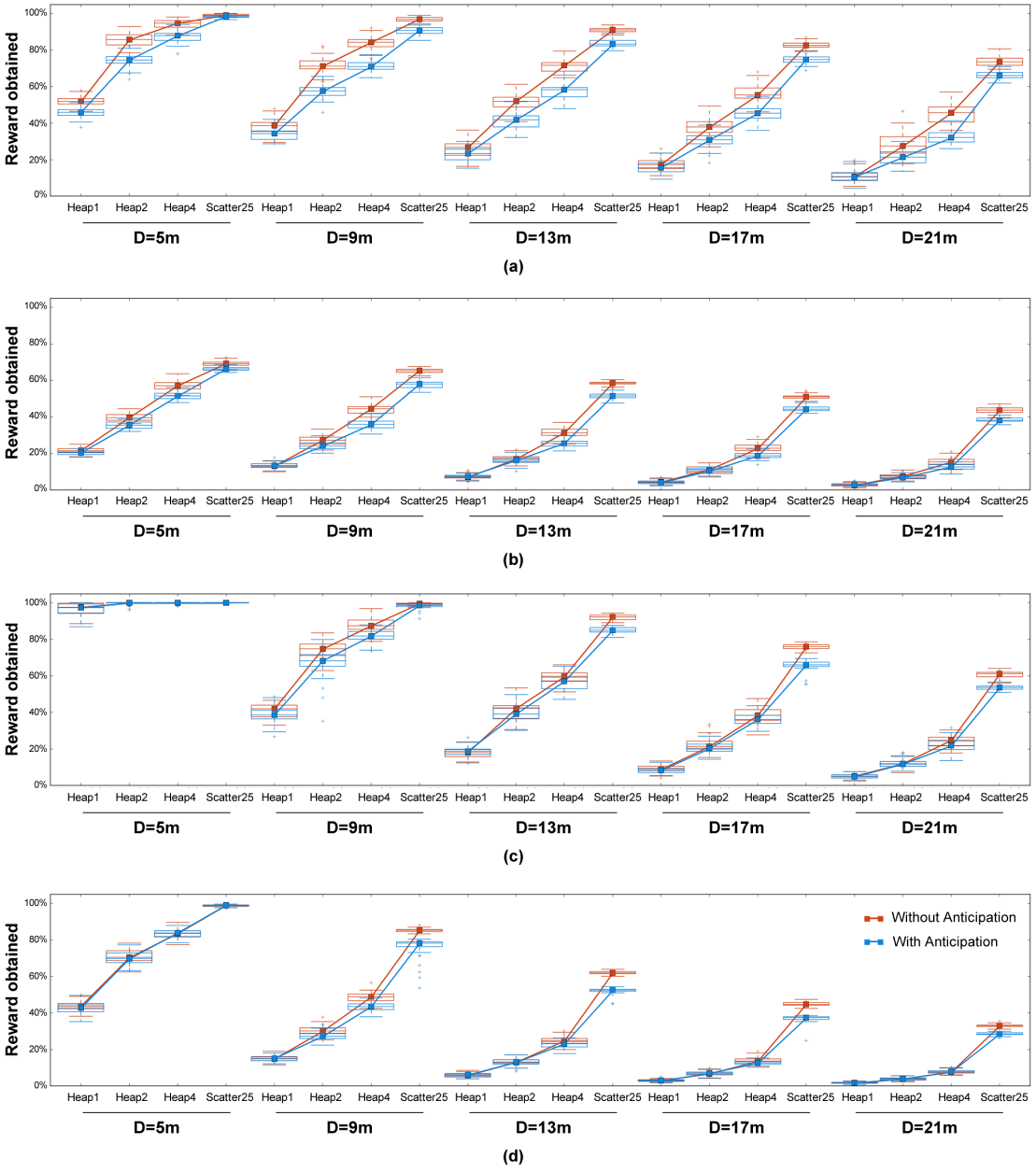


Figure G.27: The percentage of available reward obtained by 25-robot local broadcaster swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of local broadcasters was worse with Anticipation in some scenarios, especially in the slow consumption task.

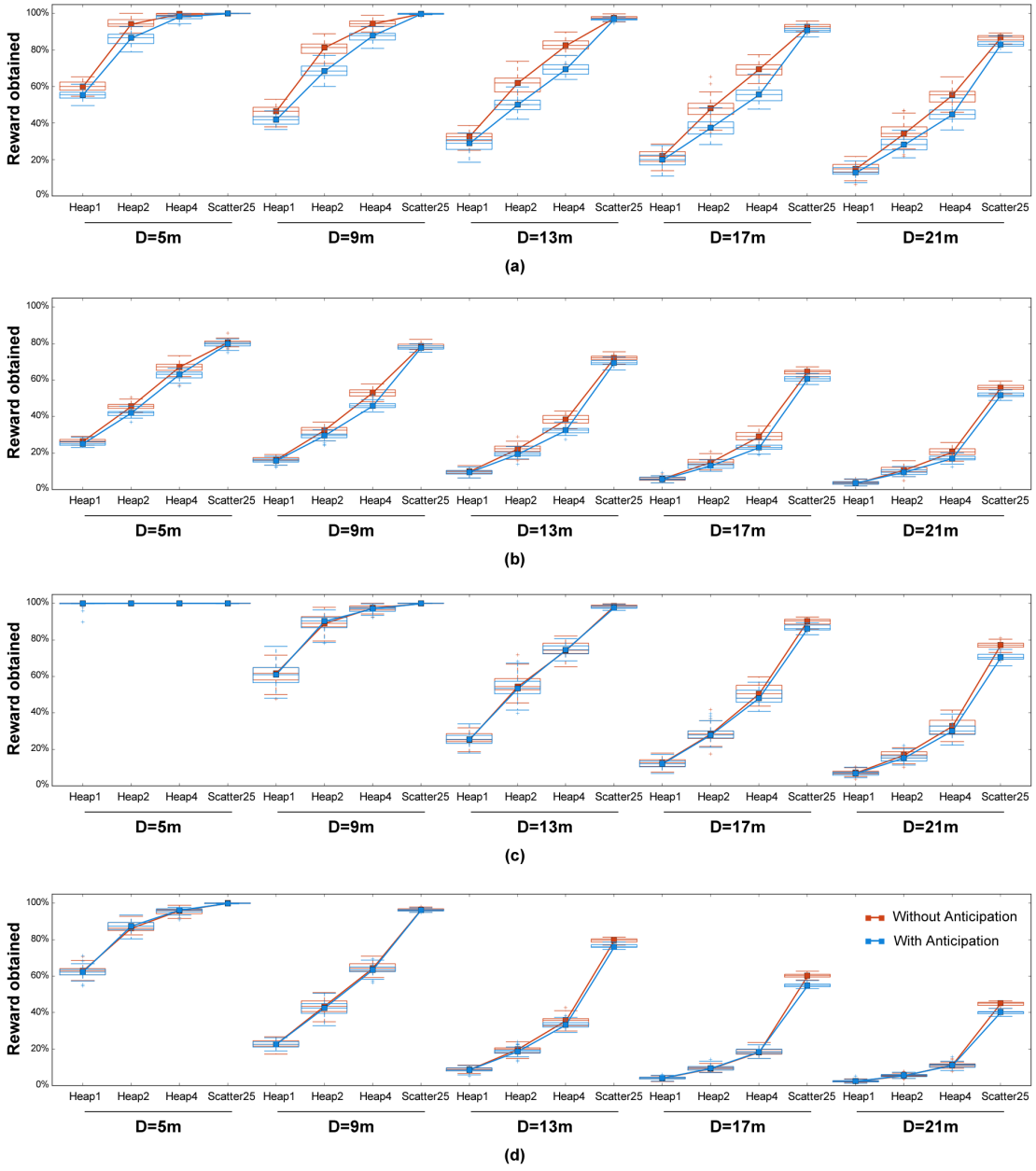


Figure G.28: The percentage of available reward obtained by 50-robot local broadcaster swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of local broadcasters was worse with Anticipation in some scenarios, especially in the slow consumption task.

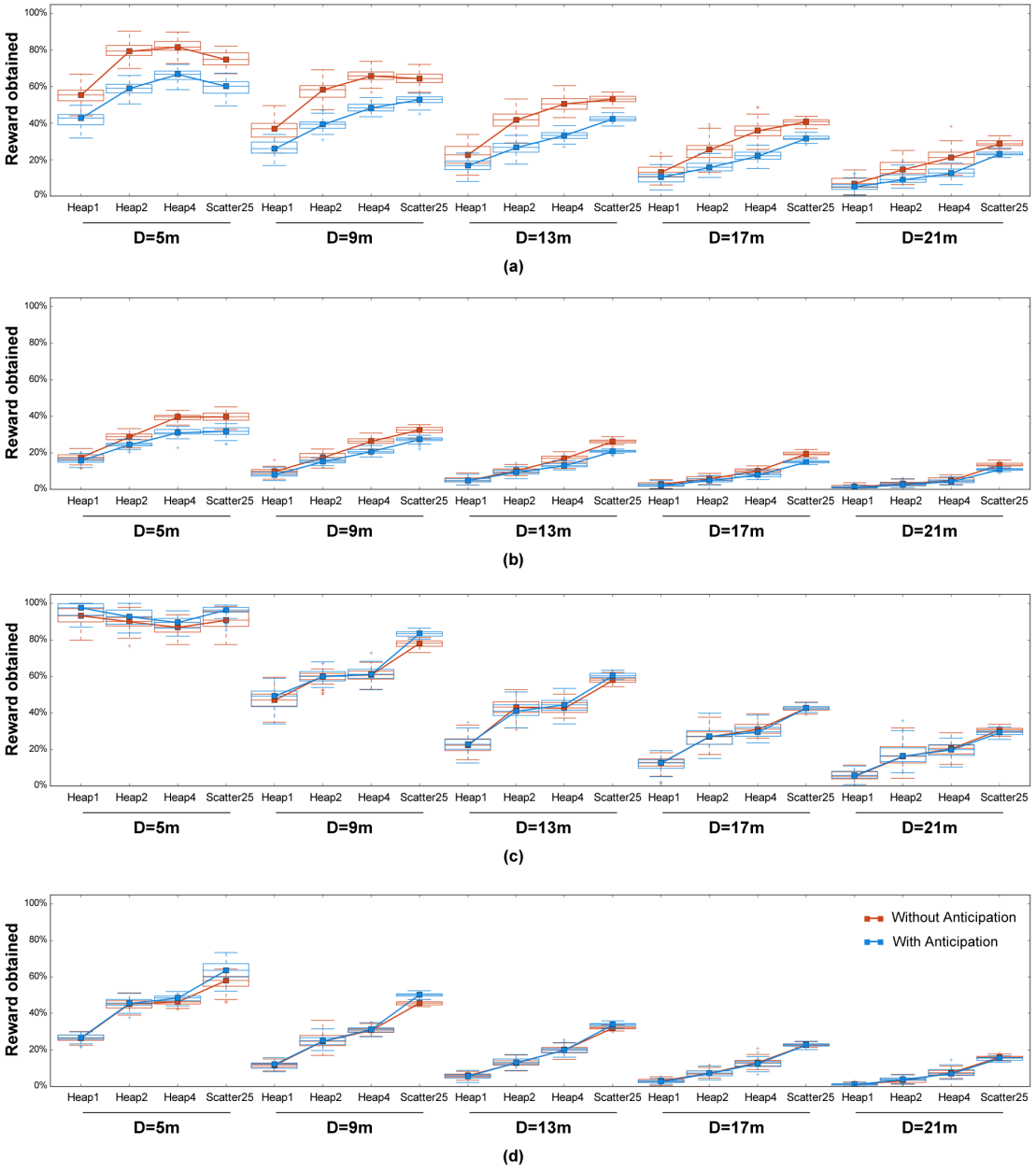


Figure G.29: The percentage of available reward obtained by 10-robot bee swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of bee swarms was worse with Anticipation in the consumption task. Anticipation increased the swarm performance in Scatter25 scenarios or when D was small during the collection task.

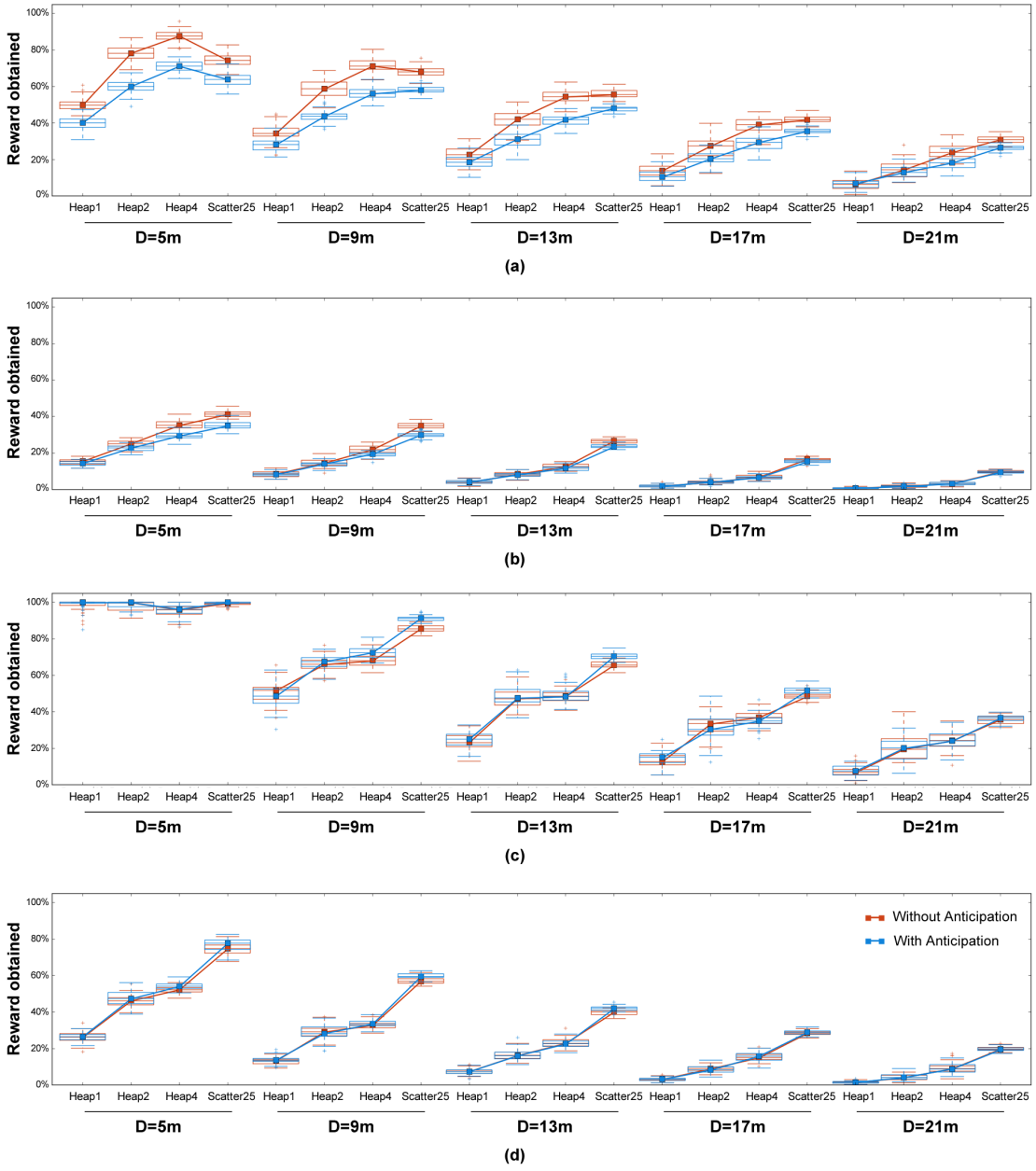


Figure G.30: The percentage of available reward obtained by 25-robot bee swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of bee swarms was worse with Anticipation in the consumption task. Anticipation increased the swarm performance in some Scatter25 scenarios during the collection task.

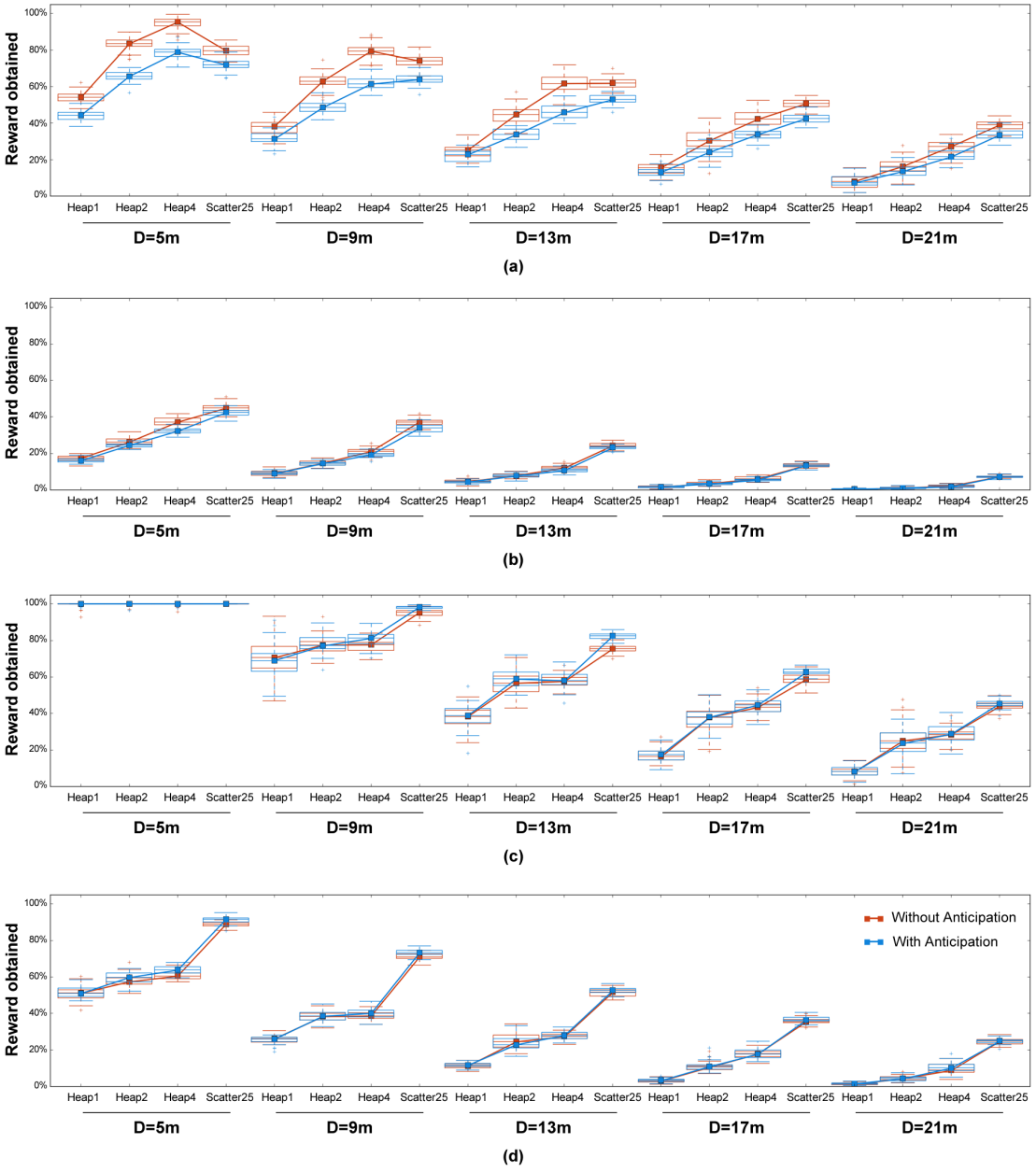


Figure G.31: The percentage of available reward obtained by 50-robot bee swarms with and without Anticipation in the (a) slow dynamic consumption, (b) fast dynamic consumption, (c) slow dynamic collection and (d) fast dynamic collection tasks. The performance of bee swarms was worse with Anticipation in the consumption task. Anticipation increased the swarm performance in some Scatter25 scenarios during the collection task.

G.5 Uncertainty cost

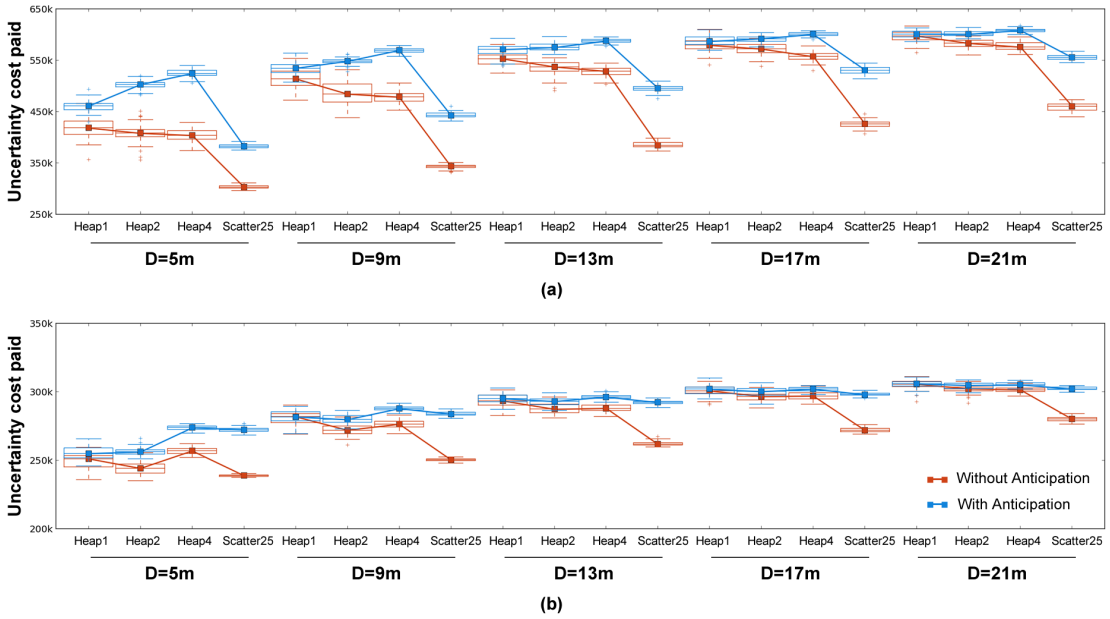


Figure G.32: Uncertainty cost, C_U, of 10-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by solitary swarms was larger with Anticipation in most scenarios.

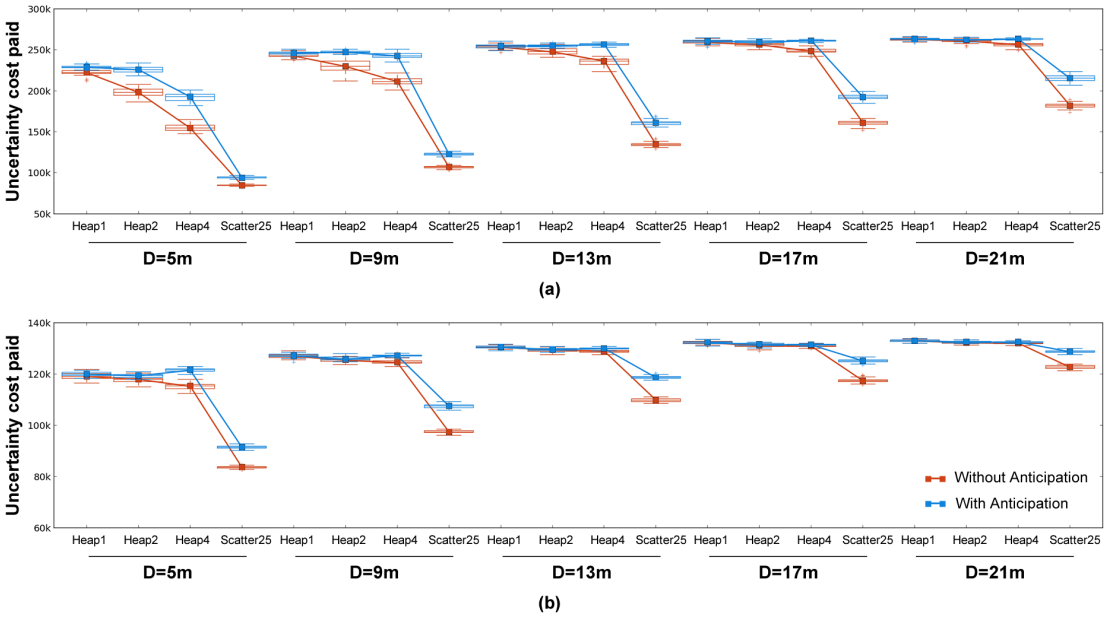


Figure G.33: Uncertainty cost, C_U, of 50-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by solitary swarms was larger with Anticipation in most scenarios.

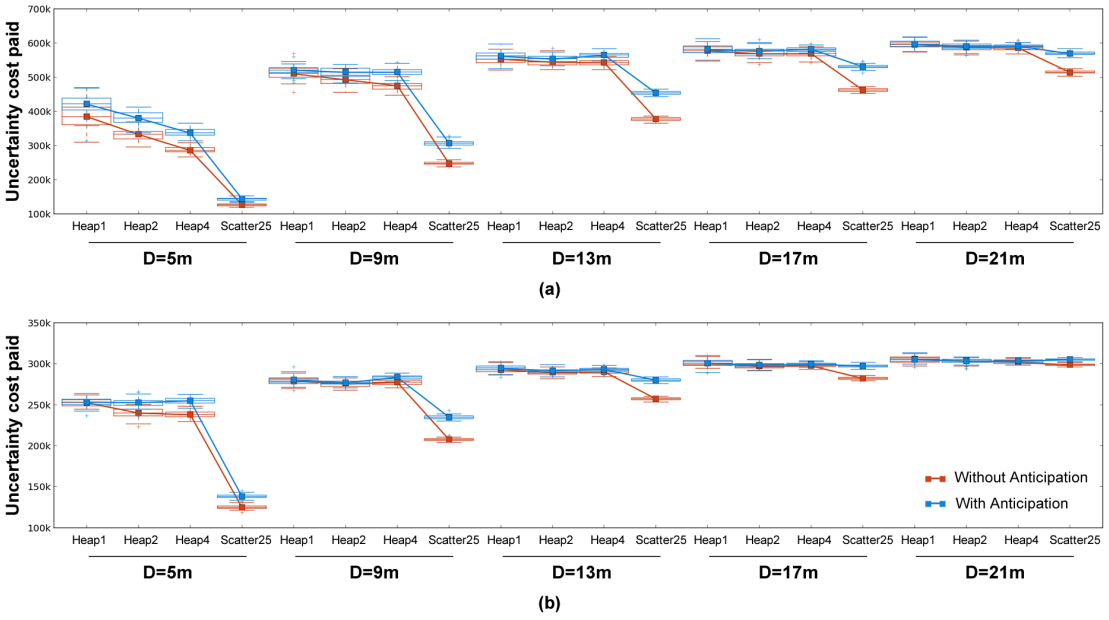


Figure G.34: Uncertainty cost, C_U, of 10-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by solitary swarms was larger with Anticipation in Scatter25 scenarios or when D was small, especially when the environment changed slowly.

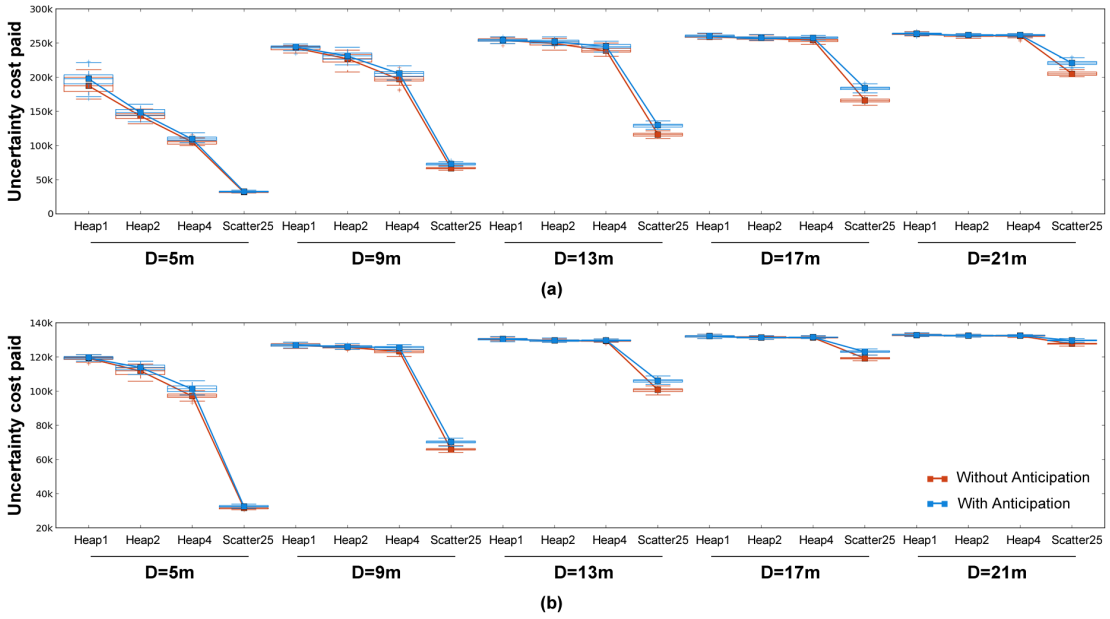


Figure G.35: Uncertainty cost, C_U, of 50-robot solitary swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by solitary swarms was larger with Anticipation in Scatter25 scenarios and some Heap4 scenarios.

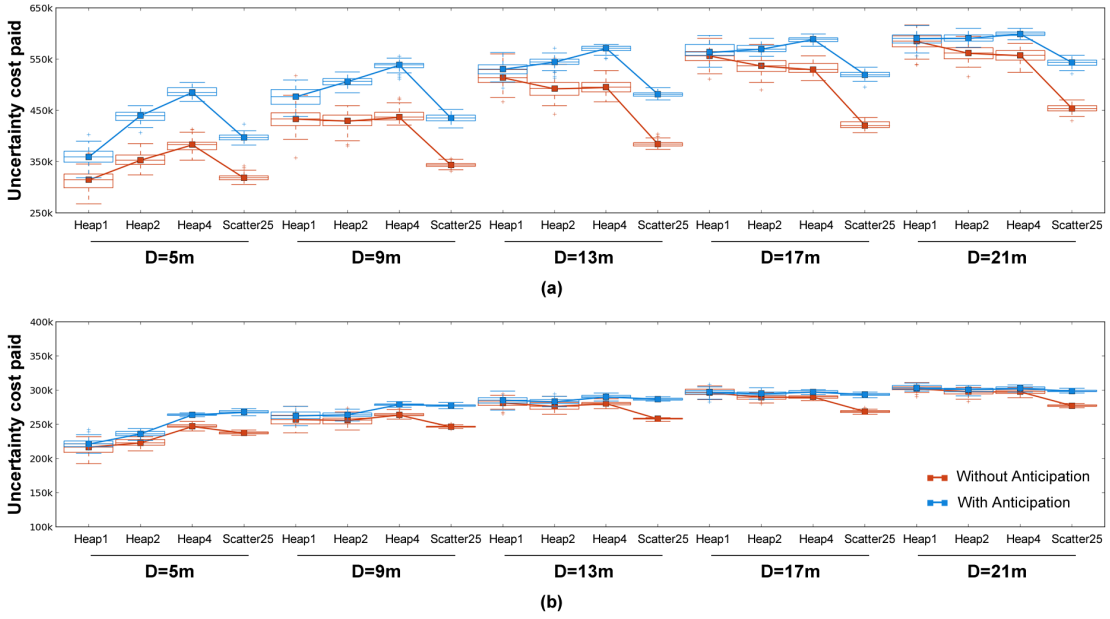


Figure G.36: Uncertainty cost, C_U, of 10-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by local broadcasters was larger with Anticipation in most scenarios.

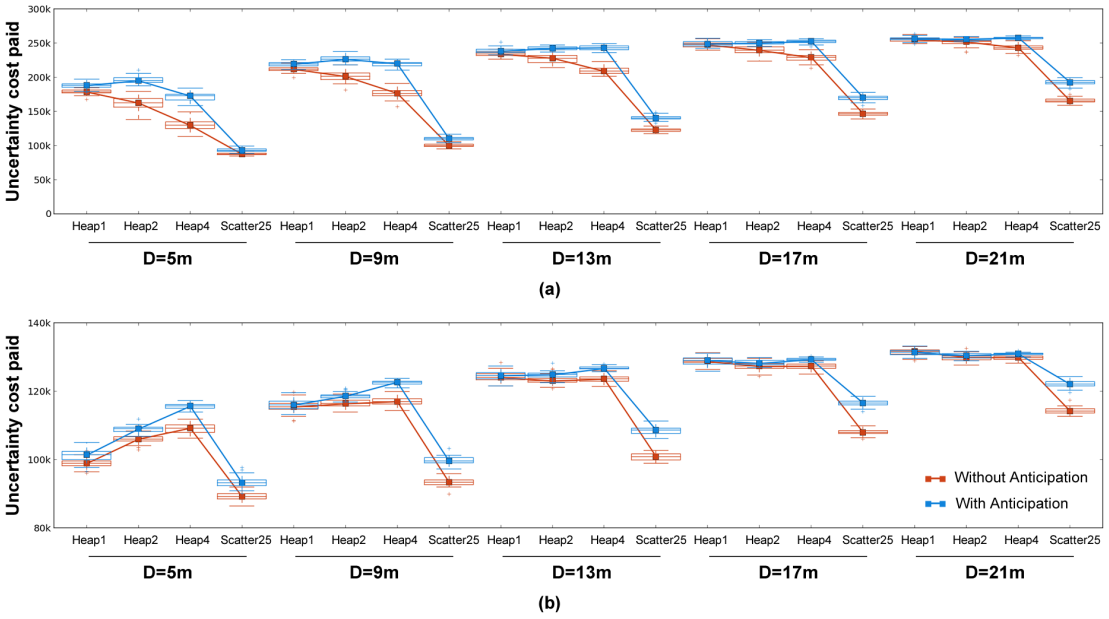


Figure G.37: Uncertainty cost, C_U, of 50-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by local broadcasters was larger with Anticipation in most scenarios.

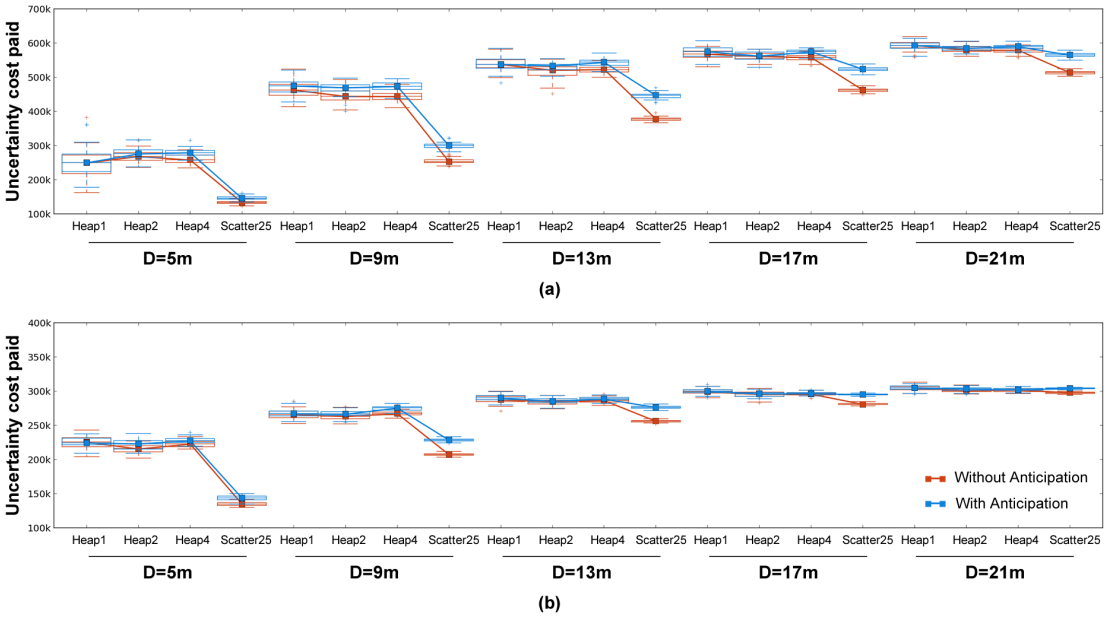


Figure G.38: Uncertainty cost, C_U, of 10-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by local broadcasters was larger with Anticipation in Scatter25 scenarios and some Heap4 scenarios.

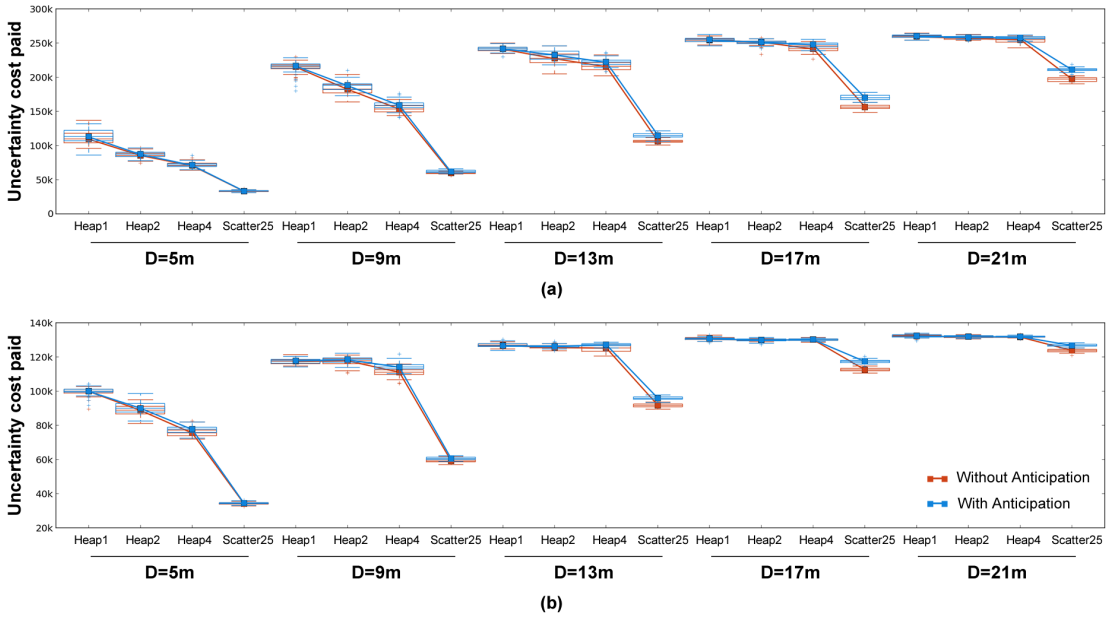


Figure G.39: Uncertainty cost, C_U, of 50-robot local broadcaster swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by local broadcasters was larger with Anticipation in some Scatter25 scenarios.

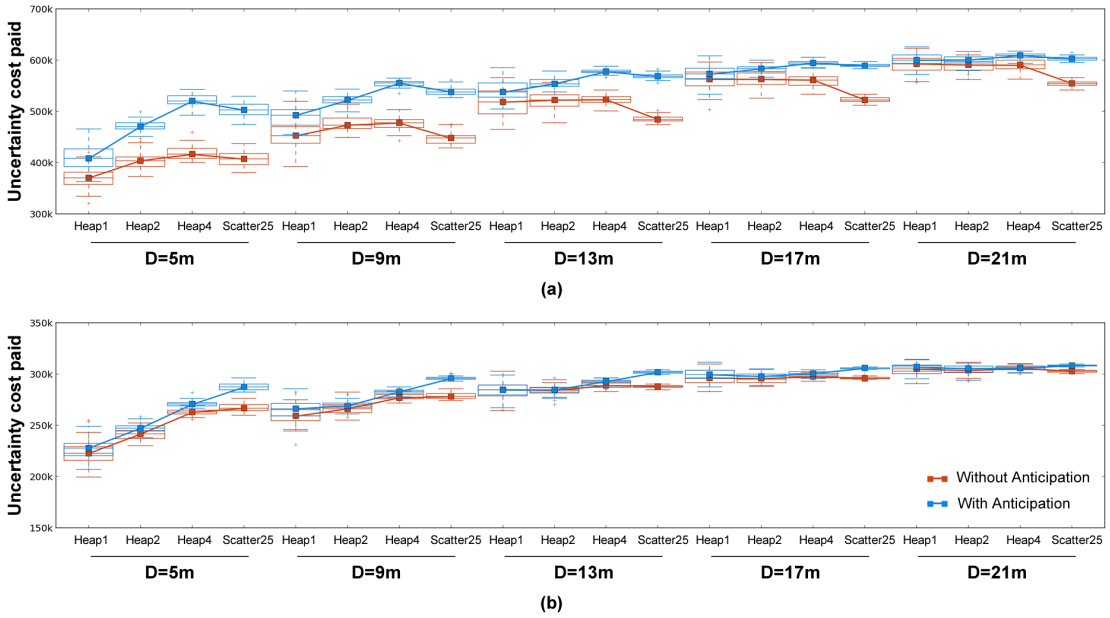


Figure G.40: Uncertainty cost, C_U, of 10-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by bee swarms was larger with Anticipation in most scenarios when the environment changed slowly. When the environment changed quickly, the amount of C_U paid was larger in most Scatter25 scenarios.

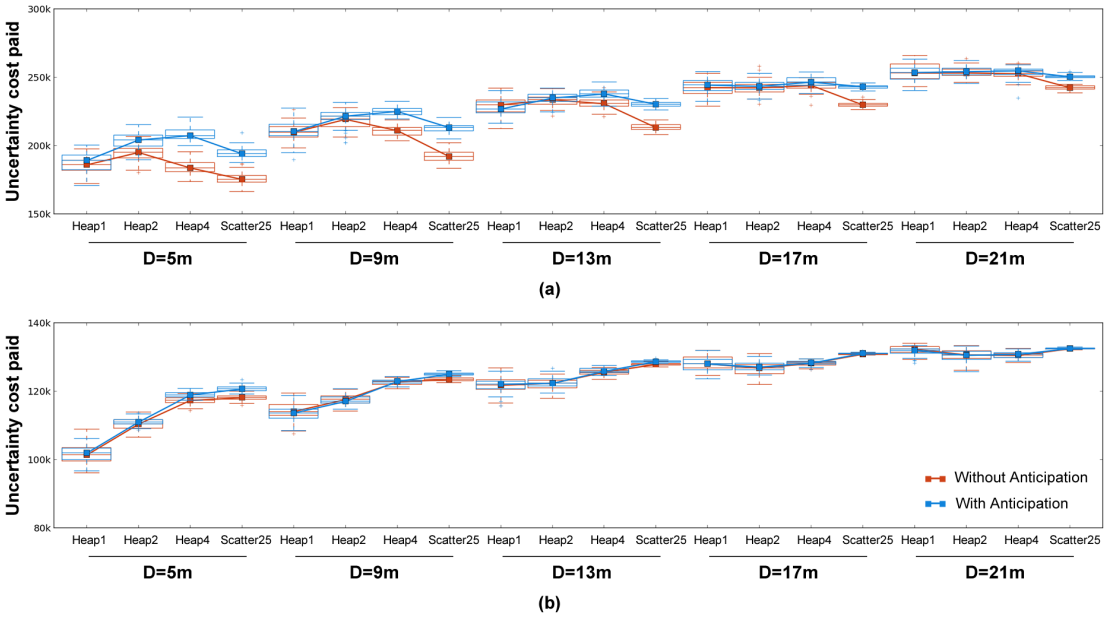


Figure G.41: Uncertainty cost, C_U, of 50-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic consumption task. The amount of C_U paid by bee swarms was larger with Anticipation in most scenarios when the environment changed slowly. When the environment changed quickly, the amount of C_U paid was larger in some Scatter25 scenarios.

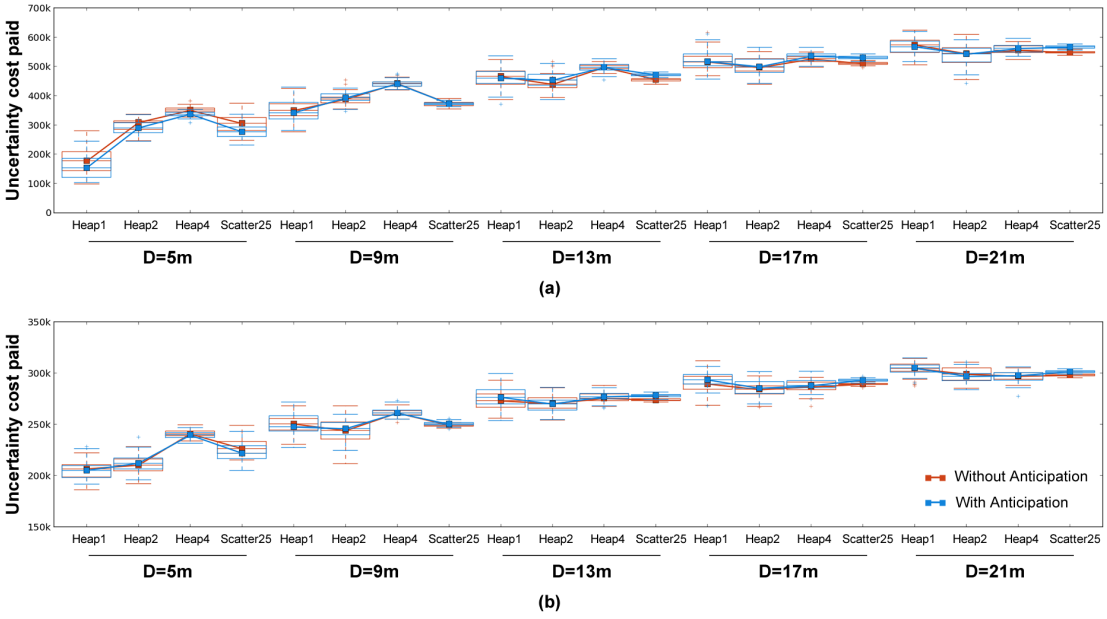


Figure G.42: Uncertainty cost, C_U, of 10-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by bee swarms is larger with Anticipation in some Scatter25 scenarios, while it remained unaffected in other scenarios.

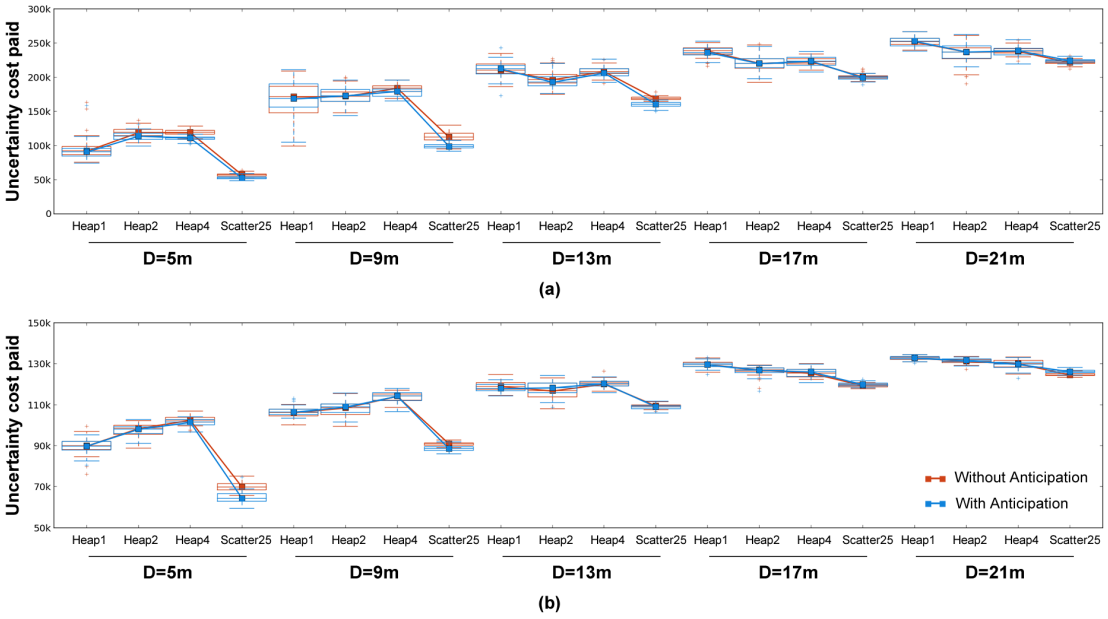


Figure G.43: Uncertainty cost, C_U, of 50-robot bee swarms with and without Anticipation in the (a) slow and (b) fast dynamic collection task. The amount of C_U paid by bee swarms is smaller with Anticipation in some Scatter25 scenarios, while it remained unaffected in other scenarios.

Appendix H

"Understanding the role of recruitment in collective robot foraging"

This paper was published as Pitonakova, L., Crowder R. & Bullock, S. (2014). Understanding the role of recruitment in collective robot foraging. In Lipson, H. et al. (eds.), *Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*, MIT Press, 264–271

Abstract

When is it profitable for robots to forage collectively? Here we compare the ability of swarms of simulated bio-inspired robots to forage either collectively or individually. The conditions under which recruitment (where one robot alerts another to the location of a resource) is profitable are characterised, and explained in terms of the impact of three types of interference between robots (physical, environmental, and informational). Key factors determining swarm performance include resource abundance, the reliability of shared information, time limits on foraging, and the ability of robots to cope with congestion around discovered resources and around the base location. Additional experiments introducing odometry noise indicate that collective foragers are more susceptible to odometry error.

H.1 Introduction

Foraging is a well-studied behaviour in both animals and robots. Just as biological species implement foraging strategies that are adapted to their specific niches, it would be desirable to tailor robot foraging strategies to the particular challenges that they face.

For example, large animals like wolves, hyenas or lions forage alone when their food is dispersed, while social insects such as ants, bees and termites recruit their nest mates to collectively obtain food from a patch that can be exploited over a number of visits. When should robots forage collectively?

Since collective strategies that rely on communication, co-ordination, interaction, etc., will tend to require more expensive robots and more programming time than independent, individualist robots, it is important for designers to resort to collective strategies only when they significantly improve collective performance.

In this work, the resource gathering performance of a simulated swarm of collective robots that recruit each other to profitable resource locations is compared with that of a swarm of individualists that forage independently. Swarms are evaluated across a range of scenarios that differ in terms of the challenge involved in locating resource deposits, as well as their spatial distribution and variation in terms of volume and quality. By analysing variation in performance we aim to uncover which environmental properties favour collective foraging approaches in robot swarms.

H.2 Background

Most animals, especially carnivores, perform solitary foraging (Gittleman, 1989). For example, forest birds search for insects alone if food is sparse and many insects are needed (Robinson and Holmes, 1982). Frigate birds searching for fish also do so solitarily, despite the fact that they live in colonies. They tend to disperse while searching in order to optimise their probability of success and rarely return to an area where they have previously fed (Weimerskirch et al., 2004). Similarly, hyenas (Holekamp et al., 2012), lionesses (Stander and Albon, 1993) and chimpanzees (Busse, 1978) hunt alone during periods of sparse prey abundance.

By contrast, some creatures forage collectively, recruiting their collaborators to profitable food locations by sharing information. Recruited foraging is most common in social insects and appears either in the form of stigmergy or direct signalling. Stigmergy is utilised by ants (e.g., Beekman, 2001) and termites (Arab et al., 2012) when they lay pheromone trails that lead to food, changing their environment such that it stores useful information that guides the behaviour of recruited conspecifics.

Bees, however, rely on directly influencing their nest mates. Successful foragers return to the nest to perform a waggle dance that communicates both food quality and location (relative to the dance floor) to watching bees (Seeley, 1994; Granovskiy et al., 2012). Flexibility and sophistication is achieved by both scouting for new food sources and re-evaluation of old foraging sites, allowing bee colonies to rapidly adapt to changing flower quality (Seeley, 1992; Biesmeijer and De Vries, 2001)

A number of approaches have been taken to implementing foraging in robot swarms, including random walking robots that forage independently (e.g., Hoff et al., 2010), bucket brigading swarms where each robot is only responsible for its own portion of a foraging area and items travel towards a base location by being passed between robots (e.g., Shell and Mataric, 2006; Lein and Vaughan, 2009), robots that directly signal to each other about where items can be found (e.g., Rybski et al., 2007; Jevtic and Gazi, 2010; Sarker and Dahl, 2011), and swarms that use stigmergy, where certain robots act as beacons, maintaining an information gradient towards a deposit (e.g., Hoff et al., 2010).

It is notable that robot swarms are often investigated in a single environmental scenario with a specific distribution of target items (typically a uniform random distribution) or in scenarios that are particularly suited to the swarm strategy. An exception to this was the work of Hoff et al. (2010) who compared a group of randomly walking individualists to a swarm that used stigmergy. However, the individualists in these experiments were highly disadvantaged as they had to find the drop off location by random walk every time they acquired a target and thus performed very poorly.

Here we implement an idealised and simplified version of bee-like recruitment in which robots that are near to each other may exchange subjective information about recently visited resource locations. By exploring a range of environments, we aim to discover both the conditions that favour collective foraging and those that do not.

H.3 Methods

H.3.1 Simulation environment

Robots of size 10×10 units were placed in a 4000×4000 continuous-space arena with periodic boundaries (i.e., a torus) featuring a centrally located circular base with a diameter of 100 units. One simulated second consisted of 50 update loops (hereafter "updates") and an experimental run lasted 600 simulated seconds. One robot travelling at top speed in a straight line could traverse the full length of the world in 80 seconds (i.e., travelling at 50 units per second).

A set of foraging scenarios were defined, each characterised by the number, N_D, of 5×5 resource deposits in the environment and their properties. Each deposit, i, comprised V_i units of material, where each unit had a deposit-specific resource quality, $Q_i \in [0, 1]$, such that the total resource in a deposit was equal to $V_i Q_i$. Each robot could extract up to one unit of volume from a resource deposit per visit, reducing the deposit's volume and extracting Q_i resource to be returned to the robot base. Once the entire volume of a resource deposit had been collected it ceased to exist in the environment and was not replaced.

By default, all deposits were distributed uniformly at random in space with each deposit having equivalent quality ($Q = 0.5$). However, some environments featured non-uniform spatial distributions comprising a number of spatially clustered *groups* of deposits, and, additionally, some environments featured non-uniform quality distributions in the form of a number of higher-quality *patches*. Note that these patches did not influence the spatial distribution of deposits, only their quality.

Where the number of spatially clustered groups, N_G, was greater than zero, each of these groups featured 10 deposits clustered within a 500-unit radius circular region and no deposits existed outside these groups. Where the number of quality patches, N_P, was greater than zero, each patch was centered on a deposit which was allocated $Q = 1$, and the quality of the deposits around it decayed with distance, d, from the patch centre:

$$Q = \min\left(e^{-d^2/\beta\sigma_P^2},\ 0.5\right), \tag{H.1}$$

where patch quality variance, σ_P^2, was a scenario-specific parameter that controlled the size of the quality patch, and β was a constant factor set to 8×10^6 for all results reported here. In such scenarios, the quality, Q, of deposits outside of patches was scaled down by a constant factor in order to keep the total amount of net resource equal across scenarios. Quality patch centres were randomly chosen deposits with no more than one patch centre per group (when $N_G > 0$).

H.3.2 Robots

The simulated robots utilised a subsumption architecture (Brooks, 1986), where low-level behavioural modules such as avoiding obstacles could be overridden by motor commands of higher-level behaviours such as navigation towards a target (Figure H.1). The individual modules communicated with each other by setting the robot's state and other internal variables stored in the robot's memory. A finite-state machine representation of the robot is depicted in Figure H.2.

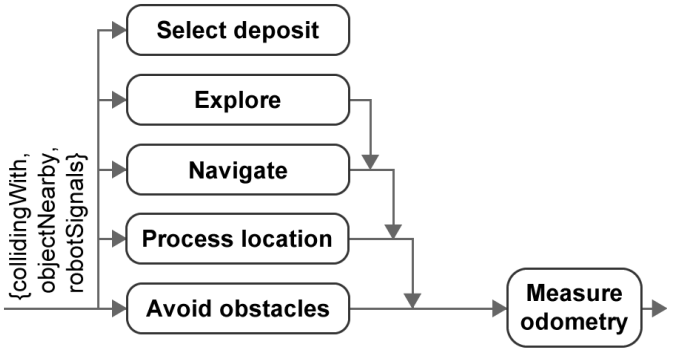


Figure H.1: Subsumption architecture of the robot controller.

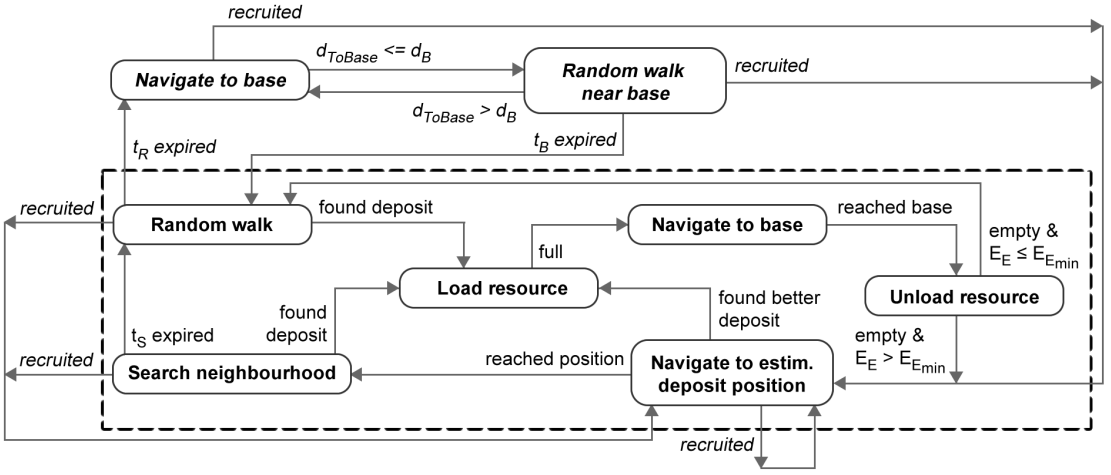


Figure H.2: Finite-state machine representation of the robot controller. B-swarm logic is shown in italics outside the dashed box.

Robots were initially randomly oriented and located within the central home base which featured a beacon that allowed any robot to navigate directly home from any point in the arena (Pini et al., 2013).

Individualist robots (I-Swarm) performed a random walk to search for food, avoiding each other by backing up and turning by a random angle. When a deposit was discovered, one unit of volume was loaded (or the entire deposit if one or less than one unit remained) and returned to the base (navigating using the home beacon). On the basis of subjective odometry calculations, a robot returned to where it estimated the deposit was located if the energy efficiency, E_e, of the deposit was better than that of the worst deposit that it had found so far.

Estimation of energy efficiency was inspired by a model of bees (Seeley, 1994) and took into account the volume, V, and quality, Q, of the deposit, as well as the cost of re-visiting it, calculated in terms of time and energy required to (i) reach it (1 unit of energy per simulation update), (ii) load it (5 seconds at 5 units of energy per update), (iii) return to base (5 units of energy per update) and (iv) unload it (2 seconds at 5 units per update):

$$E_e \sim \mathcal{N}\left(\frac{VQ - C}{C},\ \sigma_e^2\right) \tag{H.2}$$

$$C = \sum_{i=1}^{4} T_i * E_i, \tag{H.3}$$

where $\sigma_e^2 = 0.05$ was the variance of normally distributed estimation noise, and T_i and E_i were the time and energy per unit time required for each of the four foraging subtasks. It was also assumed that a robot could estimate the energy efficiency of a deposit from a maximum distance of 30 units.

Upon reaching a remembered location, neighbourhood search was undertaken for a maximum of $t_S = 10$ seconds, comprising a random walk constrained within a distance of $r_S = 120$ units around the estimated deposit location (Pini et al., 2013). The robot resumed random search if nothing was found, or if it could not reach the remembered location within 120 seconds. If, en route to a remembered location, a robot encountered a new deposit with superior energy efficiency, the superior deposit would be visited instead.

Odometry was used to estimate the location of a remembered deposit based on a robot's subjective impression of its rotational change $\Delta\alpha'$ and speed S' during locomotion. Odometry was affected by Gaussian noise on the actual values of rotation (α) and speed (S), simulating wheel slippage, etc. (Chong and Kleeman, 1997; Martinelli, 2002):

$$\begin{aligned} \Delta\alpha' &\sim \mathcal{N}\left(\Delta\alpha,\ \kappa \times \Delta\alpha\right), \\ S' &\sim \mathcal{N}\left(S,\ \kappa \times S\right), \end{aligned} \tag{H.4}$$

where parameter κ scaled the variance of odometry noise.

Collective foragers (B-Swarm) employed the same basic foraging strategy as the I-Swarm, but could also signal to nearby robots (< 60 units) their estimate of a remembered deposit's location and its associated energy efficiency (Jevtic and Gazi, 2010). A robot was recruited to a signalled location if it either did not have the location of a deposit in memory, or if its remembered deposit had inferior energy efficiency. Additionally, a B-Swarm robot returned to the base and roamed within $d_B = 200$ units of its centre for up to $t_B = 120$ seconds if it could not find a deposit during $t_R = 120$ seconds of random walk. In this way, unsuccessful foragers could discover deposit locations from robots returning to base to unload, mimicking to some extent the behaviour of bees (Biesmeijer and De Vries, 2001; Granovskiy et al., 2012).

H.4 Results

Three sets of simulation results are reported here. First, we explore the impact of varying both the number of robots and the number of deposits on I-Swarm performance in order to identify values for N_R and N_D that present an appropriate degree of challenge. Second, we compare the performance of I-Swarms and B-swarms over a range of foraging scenarios that differ in key respects. Third we explore the impact of odometry error on swarm foraging performance.

H.4.1 Calibrating N_R and N_D

I-Swarm performance was assessed in two basic scenarios, *Uniform* quality and *Patchy* quality, varying the number of robots, N_R, and the number of deposits, N_D (see Figure H.3). In both scenarios, $V = 200/N_D$, and deposits were distributed uniformly at random across the entire arena. *Uniform* quality deposits had $Q = 0.5$. The quality of *Patchy* deposits was influenced by $N_P = 30$ randomly located quality patches, each with $\sigma_P^2 = 0.01$.

In both scenarios, performance peaked for an intermediate number of robots and an intermediate number of deposits. With too few robots or too many deposits, not enough foraging trips could be carried out within the duration of a simulation. With too many robots performance deteriorated due to physical interference between foragers near the base. With too few deposits, performance was reduced by the increased challenge involved in discovering deposits far from the base.

In the *Patchy* quality scenario there was some evidence that performance depended on an interaction between N_D and N_R (influencing the degree of interference between robots) but the peak and overall shape of the performance distribution was not significantly altered. Consequently, in order to assess swarms in scenarios that varied in the challenge that they presented, and thereby avoid floor and ceiling effects, a range of values for $N_D \in [10, 100]$ and $N_R \in [10 \ldots 100]$ were established and utilised for the remainder of the results presented below.

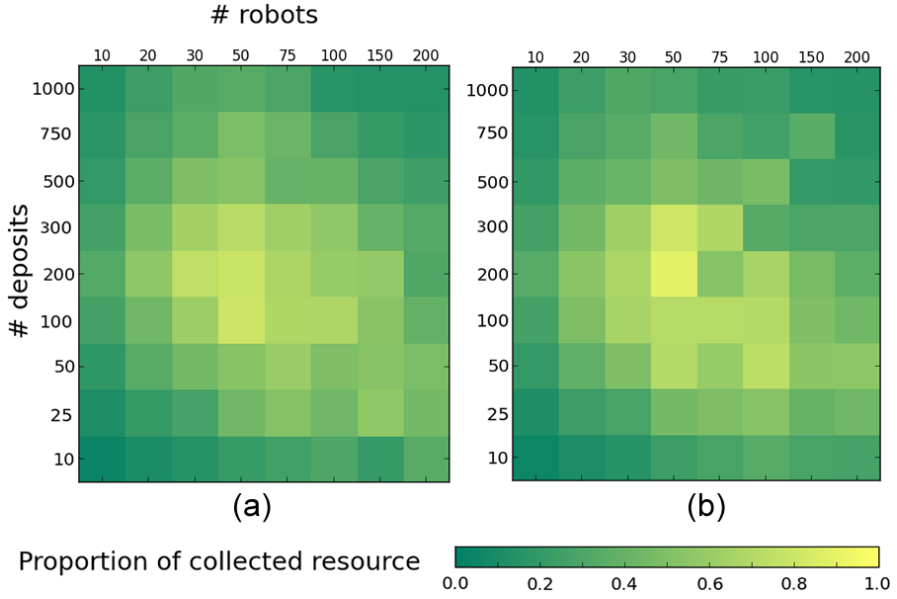


Figure H.3: I-Swarm performance as a function of the number of robots and number of deposits in (a) *Uniform* quality and (b) *Patchy* quality scenarios. Each data point is the mean of 10 independent simulations.

H.4.2 I-Swarm and B-Swarm Foraging Performance

A set of foraging scenarios were defined, varying in spatial distribution, volume, quality, etc.:

- **Litter** scattered in a field: a uniform random spatial distribution of equal quality deposits, with $N_D = 100, N_G = 0, N_P = 0, V = 2, Q = 0.5$.
- Large **Puddles** of rain water: like *Litter*, except deposits are ten times larger, $N_D = 10, V = 20$.
- Rare **Minerals**: a uniform random spatial distribution of patchy quality deposits located located at least 1500 units from the base, with $N_D = 10, N_G = 0, N_P = 5, \sigma_P^2 = 0, V = 20, Q =$ varied.
- Common **Stones**: like *Minerals*, but $N_D = 100, N_P = 10, \sigma_P^2 = 0.01, V = 2, Q =$ varied.
- **Nectar** from flower clusters: 10 groups of 10 deposits each. Location of group centres was drawn from a uniform random distribution at least 500 units away from the base, with $N_D = 100, N_G = 10, N_P = 3, \sigma_P^2 = 0.01, V = 2, Q =$ varied.
- Lost **Cargo**: like *Nectar*, but with one group of equal quality deposits where $N_D = 10, N_G = 1, N_P = 0, \sigma_P^2 = 0, V = 20, Q = 0.5$.

Variation in swarm performance across different scenarios (see Figure H.4) can be explained in terms of three types of robot-robot interference:

1. Physical interference caused by one robot obstructing another typically while foraging from the same deposit or returning to the base at the same time.
2. Environmental interference where one robot substantively alters the foraging environment of another by depleting its current target deposit.
3. Informational interference, unique to the B-Swarm, where one robot's behaviour is influenced either positively or negatively by signals from another robot.

I-Swarms slightly but consistently outperformed B-Swarms when collecting *Litter* in swarms of less than 100 robots. This advantage was statistically significant for swarms of size $N_R = 10$ and 20, and also significant across all $N_R < 100$ swarms when their results were considered together (Wilcoxon signed-rank test, $p < 0.01$). The fact that litter deposits were abundant and could be depleted in two visits meant B-Swarms suffered from informational interference that negatively impacted on performance. In theory, *Stones* presented a more significant challenge for both swarms since they were

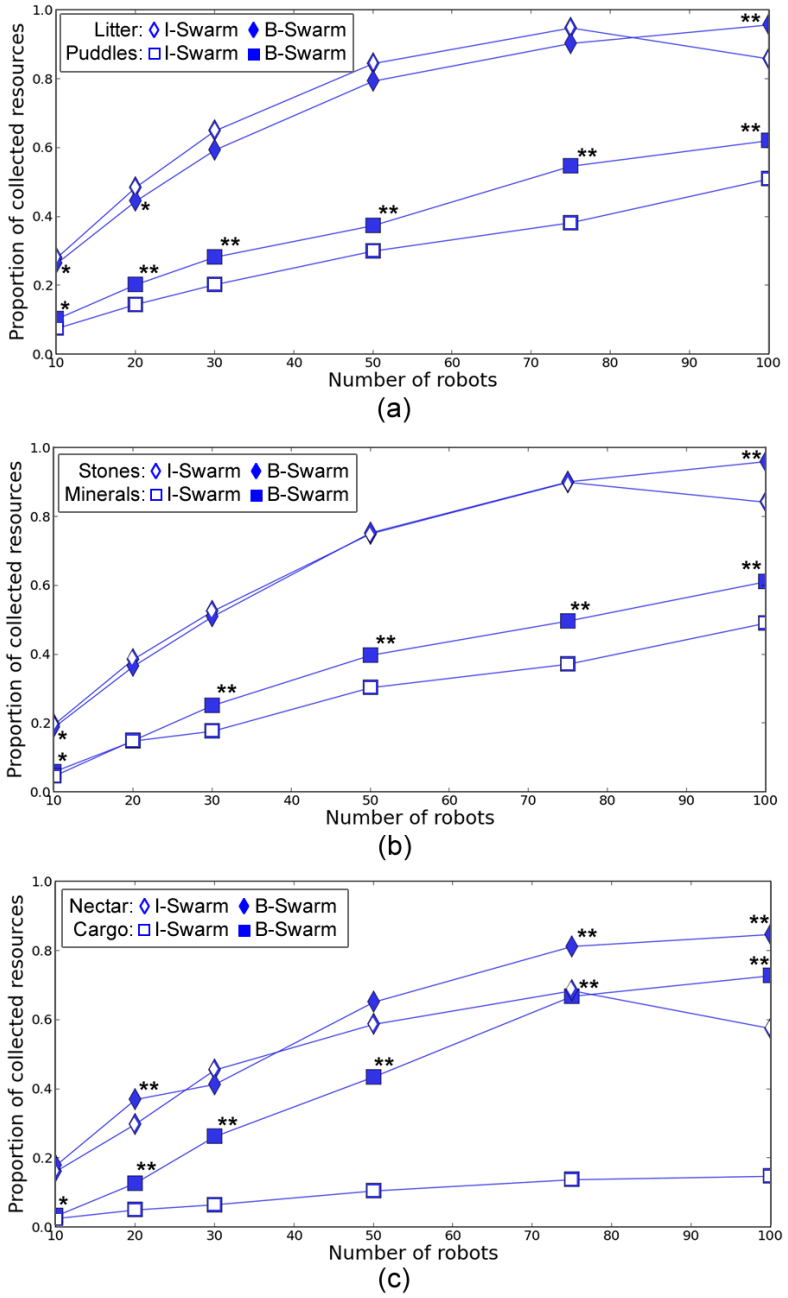


Figure H.4: The influence of scenario type on the median foraging performance of I-Swarms and B-Swarms of different sizes, based on 20 simulation runs. Statistically significant differences between I-Swarm and B-Swarm performance in the same scenario are indicated with asterisks (Wilcoxon signed-rank test, ** $p < 0.01$, * $p < 0.05$). The average interquartile range was approximately 0.082.

located further from the base and were, at least initially, harder to find. However, the slight advantage that I-Swarms experienced for *Litter* was extinguished in this scenario since B-Swarms could benefit from recruiting robots to the general location of the stones, and were thus able to match I-Swarm performance.

B-Swarms were able to collect more resources than I-Swarms in each of the other scenarios, where deposits were richer in volume but more difficult to discover, i.e., *Puddles*, *Minerals*, *Nectar* and, most strikingly, *Cargo* where all deposits were in one area far from the base. In this scenario, a B-Swarm of 100 robots was even able to collect more resource than an I-Swarm of the same size in an easier, *Nectar* collection task.

An interesting effect of recruitment was also observed when 100-robot swarms foraged for *Litter*, *Stones* or *Nectar*. I-Swarms of this size suffered a drop in performance resulting from physical interference in the form of congestion around the base. By contrast, recruitment in B-Swarms caused smaller groups of robots to forage from neighbouring locations, meaning that foragers from one group solved the problem of avoiding each other near deposits in parallel with other groups. Such self-organisation led to a kind of spontaneous traffic management and consequent reduction of near-base congestion.

On the other hand, there were cases where recruitment did not improve foraging performance. *Mineral* deposits, for instance, were very hard to find for small swarms. B-Swarms needed to have at least 30 members before it was likely that they could find a deposit early enough to ensure that recruitment enabled them to outperform an I-Swarm. Moreover, while B-Swarms typically outperformed I-Swarms when foraging for *Nectar*, this did not hold for swarms of size $N_R = 30$ and 50. Under these conditions, B-Swarms were not large enough to discover and exploit multiple groups of flowers in parallel, but were large enough to rapidly exhaust a single group of flowers by recruitment, ensuring that B-Swarms spent significant time returning to exhausted groups, and proportionately more time randomly searching rather than benefitting from recruitment. This unfortunate interplay between rapid exploitation of a group and insufficient exploration caused B-Swarms of 30 and 50 robots to perform similarly to I-Swarms of the same size.

H.4.3 Odometry error

Error in a robot's position estimation is an important influence on foraging performance where robots return to remembered deposits or are recruited to new ones. In this section, the value of the κ parameter (Equation 4), previously set to 0.005, is varied in order to explore the effect of odometry noise in the *Patchy* scenario (see Figure H.5).

Confirming the previously presented results with low odometry noise, the B-Swarm outperformed the I-Swarm when $N_D = 10$, and performed similarly or slightly worse than the I-Swarm when the number of deposits increased. Increasing κ did not affect the performance of either swarm when deposits were very easy to find ($N_D = 100$) and only slightly degraded the performance when $N_D = 25$.

The impact of odometry error was most significant when 10 deposits were placed in the environment. While collective foragers could harvest significantly more resources

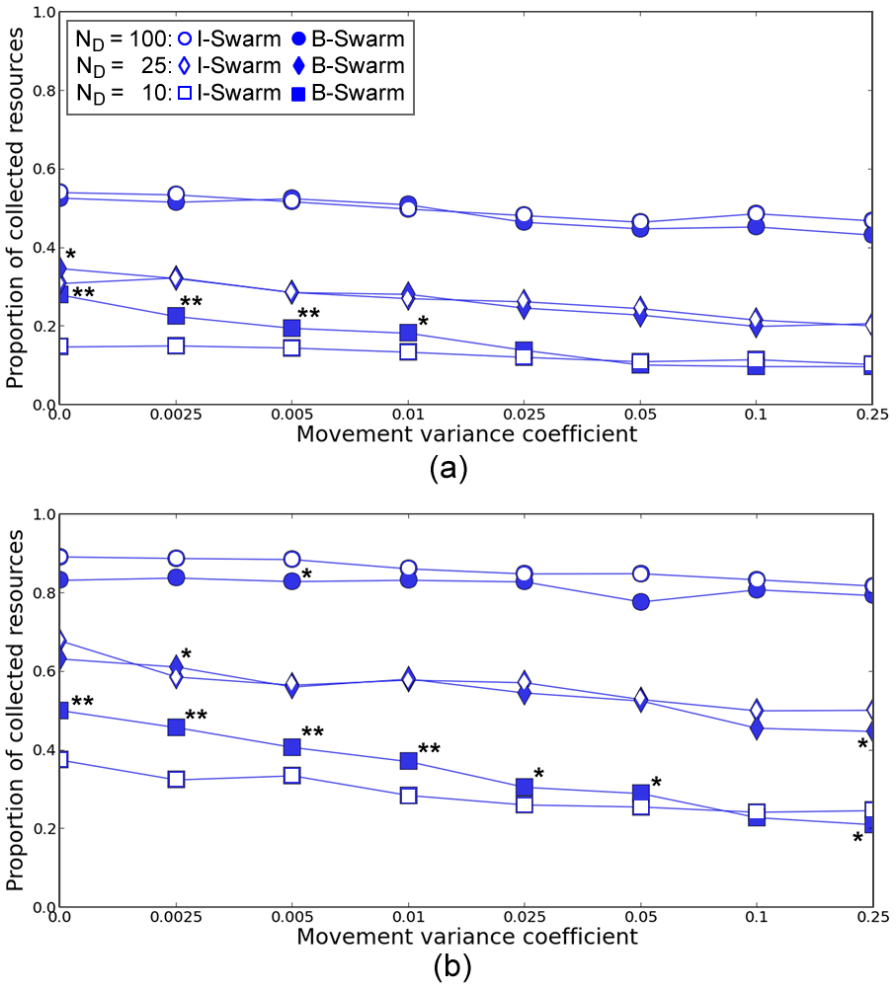


Figure H.5: The influence of odometry error (κ) on the median performance of I-Swarms and B-Swarms of different sizes in the *Variable* scenario: (a) $N_R = 20$, (b) $N_R = 50$, based on 20 simulation runs. Statistically significant differences between I-Swarm and B-Swarm performance in the same scenario are indicated with asterisks (Wilcoxon signed-rank test, ** $p < 0.01$, * $p < 0.05$). The average interquartile range was approximately 0.072.

than individualists when odometry error was low, the relative advantage of B-Swarms degraded as κ increased until it was extinguished or even slightly revered. This was an expected result, as the information about location of deposits that guided the B-Swarm robots became more erroneous with high κ, while individualist errors did not propagate to other robots in the I-Swarm, the performance of which was generally less affected by κ.

Interestingly, the impact of odometry noise on B-Swarm performance was stronger for 50-robot swarms than 20-robot swarms. While the performance of a 20-robot B-Swarm rarely fell below that of an equivalent I-Swarm even for high levels of odometry noise, 50-robot B-Swarms tended to perform worse than equivalent I-Swarms for moderate or high levels of odometry noise.

Even though robots in a 50-robot B-Swarm tended to recruit each other at a similar rate, the difficulty of finding advertised deposits when κ was high resulted in increased recruitment to deposits of low quality. For such swarms, the proportion of deposits within a higher-quality patch visited due to recruitment was 21% when $\kappa = 0.01$, but only 13% when $\kappa = 0.1$. Moreover, the distortion of information due to higher odometry error resulted in a smaller total amount of deposit collections (average of 76.9 visits per run when $\kappa = 0.01$, but only 47.1 when κ=0.1).

In contrast, while increased odometry error also negatively affected the amount of times deposits were visited by a B-Swarm of 20 robots (average of 32.5 times per run when κ=0.01; 20.6 times when κ=0.1), the smaller swarm was able to switch to more individualist behaviour when robots failed to locate advertised deposits due to poor odometry: 52% of visited deposits were found by random walk and 33% by recruitment when κ=0.01, while 63% were found by random walk and only 23% by recruitment when κ=0.1.

On the other hand, in the larger 50-robot B-Swarm it was harder to behave like an individualist when odometry failed (34% of deposit visits were due to recruitment when $\kappa = 0.01$, with a very similar 31% due to recruitment when $\kappa = 0.1$) since a bigger group generated more ultimately fruitless recruitment. If at least one searching robot eventually found a deposit, this information propagated through the swarm, causing the robots to remain near the location and try to access the resource, generating wasteful congestion instead of allowing the resumption of potentially more useful random walk exploration.

H.5 Discussion

It is often difficult to intuitively predict the aggregate results or systemic properties of swarm behaviour because of the non-trivial nature of inter-robot interactions and interactions between robots and their environment. However, the analysis presented in this paper shows that it is possible to understand swarms by studying how they behave in various environments. Here we offer a set of generalised principles extracted from the experiments.

H.5.1 When To Forage Collectively

1. When resource is hard to find: B-Swarms performed better than I-Swarms in environments where deposits were harder to find but returned more resource. In particular, the advantage of recruitment was related to the difficulty of discovering deposits (Figure H.6), but not to the length of a trip from a deposit to the base, at least within the scale of the scenarios simulated here. The B-Swarm advantage was more significant

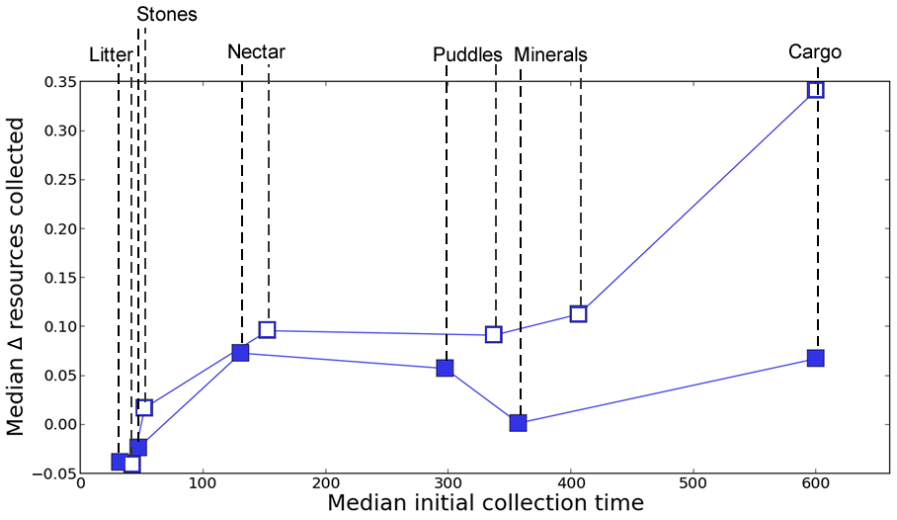


Figure H.6: The relationship between the median time taken to collect the first resource and the difference between total resource collected by a B-Swarm and an I-Swarm in various scenarios. Data is shown for 20-robot swarms (solid symbols) and 50-robot swarms (open symbols).

in larger swarms that covered the area faster and thus had a higher probability of discovering deposits. A small B-Swarm of 20 robots had difficulties, especially where deposit discoveries were rare and represented a small fraction of the total resource. These results suggest that communicating robots are useful for tasks like extracting rare minerals or retrieving lost cargo, but not for picking up litter from streets.

2. When congestion is a problem: The performance of large swarms of individualists decreased rapidly as congestion around the base prevented robots from foraging. In contrast, recruited groups of B-Swarm robots could solve congestion problems near deposits in parallel with each other, increasing the amount of robots that could operate effectively near the base at the same time. It is notable that this B-Swarm feature would become less effective if the swarm size increased significantly, as the recruited groups would become larger and there would be more of them.

H.5.2 When Not To Forage Collectively

1. When resource is abundant: I-Swarms could slightly outperform B-Swarms when items were abundant and easy to find. Not only was recruitment in these environments unnecessary, but a combination of environmental and informational interference caused advertised locations to become depleted before a recruit could reach them, ensuring that recruitment incurred a performance cost.

2. When reliability of information is low: A decrease in the reliability of information about deposit locations due to odometry error caused the foraging ability of

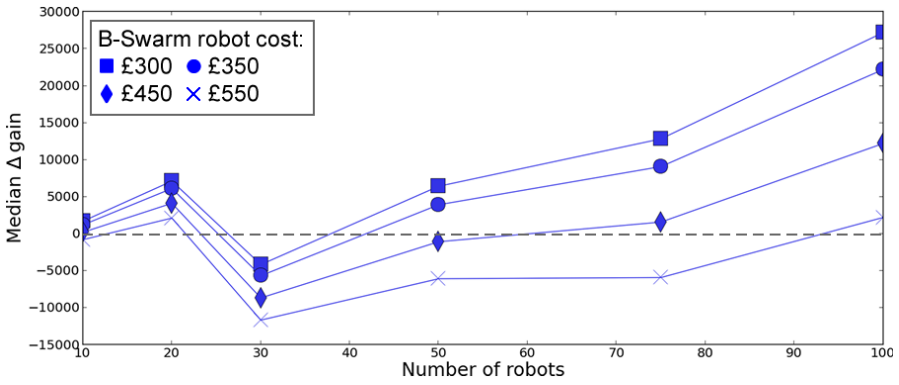


Figure H.7: Projected relative gain from using B-Swarm rather than I-Swarm to collect *Nectar*, where the cost of an I-Swarm robot is £300, the gain from collecting all resources is £100,000, and the cost of a B-Swarm robot varies.

collective foragers to degrade significantly. In these cases, I-Swarm performance may be more reliable, although still generally poor.

3. When deposits are very hard to find quickly: It was very difficult for small swarms of both types to find *Minerals* and *Cargo* where deposits were concentrated in a few distant locations. Despite the additional capabilities of the B-Swarm, once deposits were discovered there was not enough time to recruit others and exploit the location information.

4. Borderline cases: In the real world, the cost of a robot swarm must be reasonable when assessed against the value generated by the foraging task. The additional investment required in order to equip robots with communication transmitters and receivers, and program them with the extended B-Swarm logic should thus be taken into account. For example, imagine that an I-Swarm robot costs £300 and that collecting all resources from the environment is worth £100,000. If the additional cost of a B-Swarm robot is £50, the effect of such an investment is not very dramatic (Figure H.7). However, as communicating robots become more expensive, they are less able to return the extra investment.

H.5.3 Further Remarks

As currently implemented, B-Swarms were not better equipped than I-Swarms to selectively forage from more energy efficient deposits (Figure H.8). Since B-Swarm robots were recruited to any deposit with an energy efficiency higher than the lowest known E_e, a large proportion of the swarm would collect from mediocre locations instead of exploring the environment. However, it was observed that unemployed B-Swarm robots could target higher E_e deposits as a consequence of receiving information from a large number of returning foragers. It is thus possible that the ability to preferentially exploit

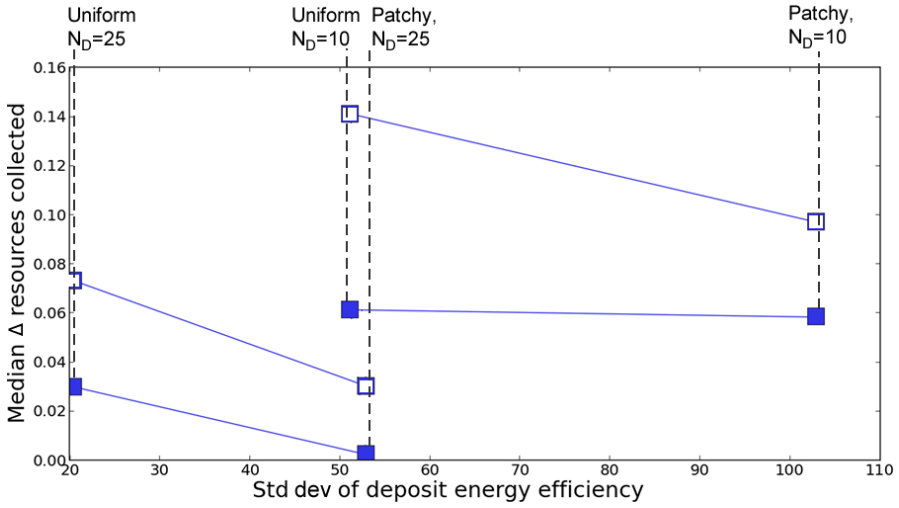


Figure H.8: The relationship between the standard deviation of deposit energy efficiency and the performance difference between B-Swarms and I-Swarms. Data is shown for 20-robot swarms (solid symbols) and 50-robot swarms (open symbols). The B-Swarm advantage did not increase in the *Patchy* variant of each *Uniform* scenario.

deposits with high E_e could be achieved by giving the robots a memory of previously encountered resources and an adaptive E_e threshold that would filter social information, as is the case in bees (Seeley, 1994). It would also be possible to implement a region of the robot base that acts as a dance floor in a beehive (Seeley, 1994; Wray et al., 2011), where robots would meet and compare advertised deposits in a more centralised fashion.

Robots designated for exploration would potentially also improve the performance of the swarm. It was shown that active scouting followed by rapid recruitment are very important factors in making bees so successful in dynamic environments (Granovskiy et al., 2012), compared to other social insects like ants (Ribeiro et al., 2009). Individual effects of the bee-inspired behaviours need to be explored in future experiments in order to establish their importance.

H.6 Conclusion

The ability of a simulated robotic swarm of individualist foragers and a swarm of collective foragers to harvest resources from various environments was compared. It was shown that recruitment is useful when resources are hard to find and are either clustered spatially or reward multiple visits, as the robots can spend relatively more time exploiting known deposits than exploring for new ones. On the other hand, the additional cost associated with building communicating robots may not be justified when there are many locations from which to gather resource or when the return expected from repeated visits to deposits is low. Furthermore, robots that rely on communication are

more susceptible to errors when positional information looses reliability as may occur through odometry error.

Despite the tempting intuition that better or more expensive swarms will deliver better results, the question of whether to equip robots with the ability to communicate is not a trivial one. Just as natural evolution shapes species to become as simple but at the same time as effective as possible within their biological niche, engineered robots can benefit from our ability to step back and consider the nature of the tasks that they face.

Appendix I

"Information flow principles for plasticity in foraging robot swarms"

This paper was published as Pitonakova, L., Crowder, R. & Bullock, S. (2016). Information flow principles for plasticity in foraging robot swarms. *Swarm Intelligence*, 10(1), 33-63. Supplementary material is available on the journal web site.

Abstract

An important characteristic of a robot swarm that must operate in the real world is the ability to cope with changeable environments by exhibiting behavioural plasticity at the collective level. For example, a swarm of foraging robots should be able to repeatedly reorganise in order to exploit resource deposits that appear intermittently in different locations throughout their environment. In this paper, we report on simulation experiments with homogeneous foraging robot teams and show that analysing swarm behaviour in terms of information flow can help us to identify whether a particular behavioural strategy is likely to exhibit useful swarm plasticity in response to dynamic environments. While it is beneficial to maximise the rate at which robots share information when they make collective decisions in a static environment, plastic swarm behaviour in changeable environments requires regulated information transfer in order to achieve a balance between the exploitation of existing information and exploration leading to acquisition of new information. We give examples of how information flow analysis can help designers to decide on robot control strategies with relevance to a number of applications explored in the swarm robotics literature.

I.1 Introduction

Plasticity in a robot swarm is the ability of the robots to repeatedly organise and reorganise in response to changing demand characteristics. As such it is central to the ability of a robot swarm to cope with real-world tasks when they are extended in time or space, since these typically feature inherent dynamism in the demand characteristics that the swarm must deal with. In this paper, we study the emergence of swarm plasticity in the context of swarm foraging with homogeneous simulated swarms that need to respond to changing resource deposits. We develop an information flow measure that allows us to understand the swarm's self-organisation and present robot behavioural strategies that lead to plasticity at the level of the entire swarm.

Swarm foraging is a behaviour that has received a considerable amount of attention in swarm robotics (e.g., Krieger and Billeter, 2000; Liu et al., 2007a; Gutiérrez et al., 2010). Robots must find and collect resources distributed throughout their environment. While designing strategies for swarm foragers is challenging, such strategies allow swarms of robots to co-ordinate their exploitation of the environment and to potentially forage more effectively. Furthermore, foraging is often used as a paradigm for studying other swarm behaviours, such as task allocation (e.g., Jevtic et al., 2012) or labour division (e.g., Zahadat et al., 2013), where worksites need to be searched for in the environment and work is distributed among the robots, or agent dispersion (e.g., Ranjbar-Sahraei et al., 2012), where robots need to coordinate their movement via communication. We choose foraging as a testbed application in this paper so that our results are relevant to a spectrum of topics covered in the current swarm robotics literature.

If we could create swarms that are not only able to perform given tasks, but to perform them autonomously and reliably over a prolonged period of time in changeable environments, robot swarms could collect garbage from our streets, extract and gather minerals or be used in logistic applications. However, it is not their ability to forage itself, but rather their ability to adapt that makes swarm foraging systems interesting and useful (e.g., Dai, 2009; Ducatelle et al., 2011). Therefore, if robot swarms are to address real-life applications, we need to understand how swarm intelligence works in dynamic environments. In order to do this, we need to establish ways of analysing artificial swarms so that we can generalise findings from particular experiments and aim towards a framework or a set of design principles that can guide future research and engineering. Recent work on self-organisation in artificial swarm systems has shown an increasing demand for such generalisations (e.g., Parunak and Brueckner, 2004; Serugendo et al., 2006; Winfield, 2009).

In order to gain a detailed understanding of swarm foraging and identify what robot-level behaviour leads to swarm-level plasticity, we simulate and analyse foraging robot swarms of a gradually increasing complexity in a number of different environments. Our robots utilise recruitment, i.e., they inform each other about foraging locations through direct

communication. Section I.2 gives an overview of relevant aspects of social insect foraging behaviour and discusses state of the art in swarm robotics. We introduce our simulation environment and the robot control algorithm in Section I.3 and evaluate performance of our robot swarms in simple static environments in Section I.4. We then test the ability of swarms with various communication strategies to discover and discriminate between deposits of different quality in Sections I.5 and I.6, and we demonstrate the importance of a balance between exploration and exploitation in dynamic environments, where deposit quality changes over time. By measuring *information value*, which captures the direct effect of information flow on resource collection, we show in Sections I.7-I.9 that if information spreads slowly through swarms they are likely to remain effective as foraging conditions change, due to their ability to balance information utilisation and acquisition. On the other hand, if information spreads quickly through swarms, while performance may be improved under specific environmental conditions, such swarms may also be less capable of plastic self-organisation. The information value measure can thus help designers to decide on robot behavioural parameters that lead to desired work modes for their swarms. Furthermore, in Section I.9, we offer a set of swarm design principles focused on communication and discuss their relevance to a wider range of problems including swarm foraging from multiple sites in parallel, swarm self-regulation, emergent task allocation and evolutionary robotics.

I.2 Background

I.2.1 Swarm foraging in nature

The behaviour of swarm robot foragers is often inspired by ants or bees. Colonies of these social insects demonstrate an incredible ability to self-organise when their environment changes and have therefore been studied by both biologists and engineers.

Ants use pheromone trails to indicate paths through the environment and to inform their nest mates about foraging locations (Sumpter and Beekman, 2003; Arab et al., 2012). Pheromone paths that lead to better food sources and are used by more ants become stronger over time and attract more and more workers, while paths to inferior sources, that are not being reinforced frequently enough, gradually evaporate.

Bees are another example of animals that use recruitment when foraging. However, unlike ants, bees use direct signalling and have a designated area in the hive, called the *dance floor*, where they perform recruitment *waggle dances* for nest mates that are interested in foraging (von Frisch, 1967; Seeley et al., 1991; Biesmeijer and De Vries, 2001). While the length and strength of a waggle dance is related to the quality of a particular flower patch, the position and orientation of a bee on the dance floor relative to the sun encodes the location of a patch relative to the hive, allowing recruits to travel

to specific advertised food sources. An individual's decisions about whether to waggle dance, forage or abandon a patch are affected by olfactory and taste information from nectar samples obtained from other foragers through *trophallaxis* after one bee sends a *begging signal* to another (De Marco and Farina, 2003; Farina et al., 2005). For example, when a forager discovers that other bees are processing nectar of a much better quality, it abandons its own source faster and has a higher preference for the better source for a number of days (De Marco and Farina, 2001).

Compared to ants, bees are generally better at achieving plasticity. For example, ants find it difficult to establish a new shorter route to a food source if an established trail already exists (Ribeiro et al., 2009). Furthermore, ants rely on pheromone evaporation, which takes time and makes diversion of foraging effort from a depleted to a new patch relatively slow. On the other hand, bees abandon patches based on individual decisions, making their responses faster when a patch is depleted (Sumpter and Beekman, 2003). Bee colonies also have the ability to switch between patches when their relative quality changes (Seeley et al., 1991). It has been argued that trophallaxis plays an important role in dynamic environments, as it allows information about flower patch quality to spread through the whole hive within hours, while waggle dancing only affects bees that follow dances and is thus a slower communication method (Farina et al., 2005). Additional evidence suggests that returning scouts use *stop signals* to directly inhibit waggle dances of recruiters that advertise alternative sites (Seeley et al., 2012). Bee colonies also achieve flexibility through opportunistic scouting when a recruited forager gets lost due to errors in signal propagation during waggle dance (Seeley, 1994) and through *inspection*, i.e., occasional re-evaluation of abandoned flower patches (Granovskiy et al., 2012).

I.2.2 Robot swarm foraging in static environments

There is a considerable volume of research that concentrates on swarm foraging during which robots are solitary and do not communicate at all. Such swarms are often used in static environments to retrieve targets and bring them back to a designated location (e.g., Arkin, 1992; Balch, 1999; Ulam and Balch, 2004).

Ant-inspired robots in simulated experiments can drop cues directly into the environment in order to help others reach items of interest (Drogoul and Ferber, 1993). The use of pheromone and its evaporation and dispersion can also be simulated (Fujisawa et al., 2014). In the real world, robots that move on a phosphorescent floor can use LEDs to create glowing paths (Mayet et al., 2010). Alternatively, robots can deposit alcohol trails and use chemical sensors to follow them (Russell, 1999; Fujisawa et al., 2014) or a centralised server can store virtual pheromone deposited by robots and use a projector to display it, allowing robots to follow pheromone trails using visual sensors (Kazama et al., 2005; Garnier et al., 2007). In order to avoid the difficulty of using pheromone-like substances that require a specific arena setup, a virtual pheromone is often represented

by designated stationary robots that communicate pheromone levels to others nearby (e.g., Hoff et al., 2013; Ducatelle et al., 2011). Similarly, in a setup inspired by bee trophallaxis, a portion of the swarm is designated for propagation of values passed from robot to robot, allowing a gradient to be established between the base and a resource patch (Schmickl and Crailsheim, 2008; Nouyan et al., 2009). It is also possible to use the whole swarm as a medium that holds pheromone paths, allowing virtual ants to travel through the robots and establish the shortest path to a resource (Campo et al., 2010).

Some ant-inspired (e.g., Krieger and Billeter, 2000), as well as bee-inspired (e.g., Alers et al., 2011; Lee and Ahn, 2011) robotic systems use direct signals that allow robots to recruit others to specific deposit locations when they arrive at the base. Alternatively, robots may communicate at any point when they meet each other during foraging (e.g., Valdastri et al., 2006; Gutiérrez et al., 2010; Miletitch et al., 2013). To localise themselves and the objects of interest, robots use path integration and usually store one vector pointing towards the base and one pointing towards a found deposit.

Bee-inspired algorithms have also been applied to help robots aggregate in areas of interest (Schmickl and Hamann, 2010) or to optimise a swarm's energy intake from resource that was collected in the environment and processed in the base (Thenius et al., 2008). It is often difficult to perform experiments with large swarm sizes, to repeat experiments a sufficient number of times, or to collect statistics about robot behaviour during real-world experiments, causing many researches to rely on simulated data when a thorough analysis of swarm behaviour is required (e.g., Lee and Ahn, 2011; Liu et al., 2007a; Campo and Dorigo, 2007; Miletitch et al., 2013).

I.2.3 Robot swarm foraging in dynamic environments

When foraging in environments that change over time, swarms need to possess some form of self-organisation on the level of the collective or adaptation at the individual level in order to cope with changing foraging conditions. Bees use a combination of both techniques. They adapt their response thresholds to various stimuli (Seeley, 1994) and learn odours of profitable flowers (Farina et al., 2005). Colonies are also capable of self-organisation that results from evolved responses of bees to individual and social information (e.g., Seeley et al., 1991; De Marco and Farina, 2003).

A frequently studied swarm robot behaviour is the ability of swarms to adjust the number of foragers and resting robots based on changing resource abundance. Robots can be equipped with means to perceive their own foraging performance and the performance of others and adapt their own control parameters accordingly (e.g., Campo and Dorigo, 2007; Liu et al., 2007a). Alternatively, self-organisation on the swarm level, that emerges when robots change their actions based on individual and social information, can also be used to make swarms adapt (e.g., Sarker and Dahl, 2011).

In other experiments, foraging robots were required to distribute their foraging effort between different types of resource proportional to type abundance. It was shown that self-organising swarms could successfully solve this task if a correct balance between information sharing among robots and information acquisition by robots is achieved (e.g., Jones and Mataric, 2003; Schmickl et al., 2007).

Finally, some foraging experiments involve bucket-brigading, where robots form chains of adjacent "working areas" that run from the deposit location to the robot base, and robots move resources along these chains (e.g., Shell and Mataric, 2006; Nouyan et al., 2009; Pini et al., 2013). An individual robot's working area can be adapted on-line to give a swarm the ability to follow mobile deposits (Lein and Vaughan, 2009).

Here, in an approach similar to that of Jones and Mataric (2003); Schmickl et al. (2007); Sarker and Dahl (2011), robots do not adapt their individual control parameters, but share information with one another in order to achieve self-organisation at the level of the swarm.

I.3 Methods

I.3.1 Simulation environment

All the experiments reported here are performed in the ARGoS simulation environment (Pinciroli et al., 2012) using MarXbot robots (Bonani et al., 2010). ARGoS is a C++ environment with a realistic 3D physics engine that was specifically designed for program compatibility with MarXbots (also called foot-bots) and e-pucks. By using this type of simulation environment, we can model not only decision making of robots, but also their physical interactions with each other and with the world. The physical aspect is important as it can lead to various interferences between robots and affect their performance, for example when multiple robots try to access a deposit at the same time or when they attempt to communicate with each other (Pitonakova et al., 2014).

The simulation takes place in continuous space and updates itself 10 times per second. The foraging arena is 50 m $\times$ 50 m large and contains a centrally located circular base surrounded by resource deposits (Figures I.1 and I.2). A similar setup has been used previously in simulated (e.g., Balch and Arkin, 1994; Campo and Dorigo, 2007) and real world (e.g., Labella et al., 2004; Gutiérrez et al., 2010) experiments.

The base has a radius of three metres and it is divided into two sections: an interior circular dance floor and an annular unloading area around it (Figure I.1). A light source is placed above the middle of the base that the robots can use as a reference for navigation towards and away from the centre of the base (as in, e.g., Krieger and Billeter, 2000; Ferrante and Duéñez-Guzmán, 2013; Pini et al., 2013).

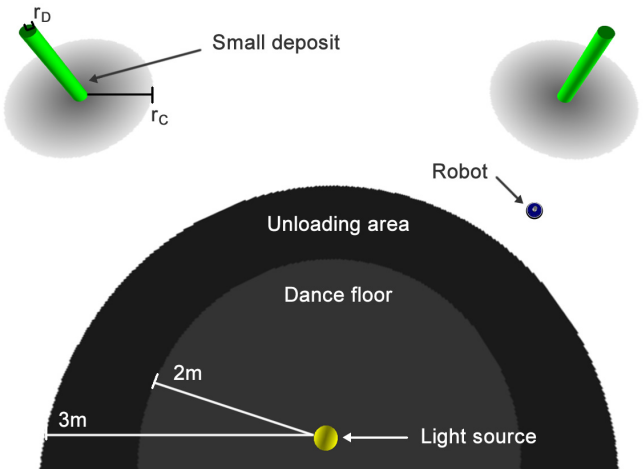


Figure I.1: ARGoS simulation screenshot of a base and scattered small deposits. The base consists of a circular dance floor with a radius of 2 m, where recruitment takes place, and an unloading area that forms a ring around the dance floor where returning foragers drop collected material. There is also a light source above the centre of the base. The whole base has a radius of 3 m. A robot with radius of 8.5 cm is located near the base. Each deposit has a colour gradient around it to allow nearby robots to navigate towards it.

Resource is distributed throughout the environment in the form of a number, N, of discrete deposits. Each deposit is cylindrical, with radius, r_D. The value of deposit i to a robot is related to two properties, the *volume* of the deposit, V_i, which represents the gross amount of material it contains, and the *quality* of the deposit, Q_i, which represents how rich this material is. The total amount of resource in a deposit is thus $V_i \times Q_i$. Each robot loads a maximum volume of $L_{\max}$ units per foraging trip. The total amount of resource that a robot may load from deposit, i, in a single trip is thus $L = L_{\max} \times Q_i$ or $L = V_i \times Q_i$ if $V_i < L_{\max}$.

Consider two deposits, A and B, with volumes greater than $L_{\max}$ and where deposit A has twice the volume of B, but deposit B has twice the quality of deposit A. Robots carrying all of deposit A back to the base will have foraged the same amount of resource as robots carrying all of deposit B back to the base because $V_A \times Q_A = V_B \times Q_B$, but robots foraging from deposit A will have needed more trips to achieve this since $V_A > V_B$.

In the following environments the volume of each deposit is $100/N$ units, i.e., total volume of deposits is 100 units and is conserved across environments. N is always selected so that V_i is an integer. The default maximum value of L is $L_{\max} = 1$ unit, although $L_{\max}$ is reduced to 0.25 units in later simulations in order to increase the amount of time taken by swarms to complete the foraging task. The default value of Q for all deposits is 1, although some scenarios are specifically designed to contain deposits of varying quality.

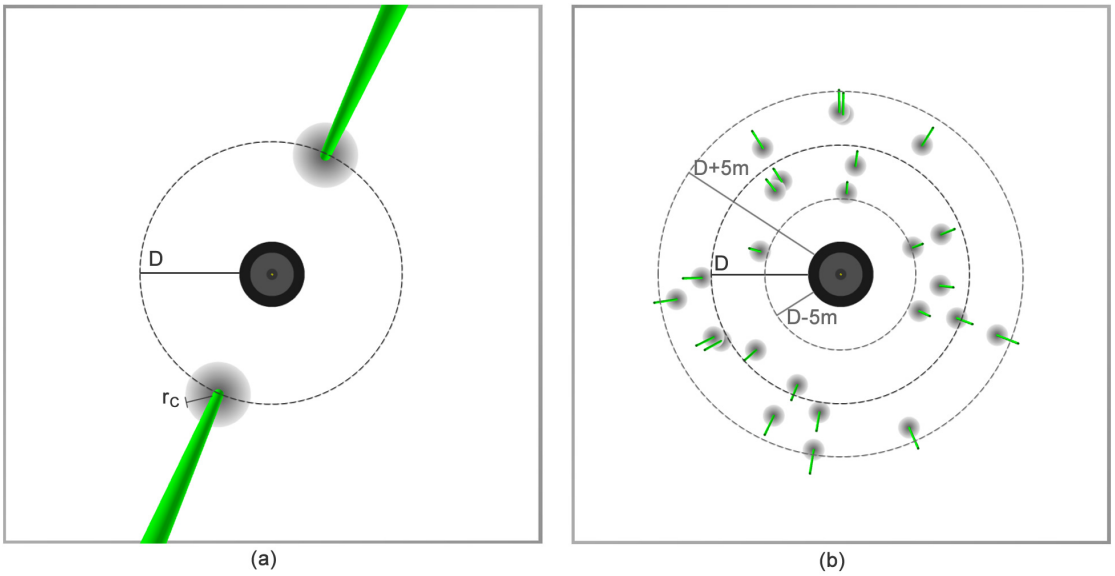


Figure I.2: ARGoS simulation screenshot of the experimental arena containing a base in the centre and deposits in the (a) Heap2 and (b) Scatter25 scenarios with $D = 9$ m. The deposits are represented as cylinders, with their heights corresponding to their respective volumes, and have colour gradients with radius r_C around them to guide navigation of nearby robots.

In order to enable robots close to a deposit to move towards it, a colour gradient with radius r_C is centred on the floor around each deposit. In a real-world experiment, navigation based on the colour gradient could be replaced by visual-based navigation, for instance.

There are two types of scenario (Figure I.2):

- **Heap$N$:** N deposits distributed evenly around the base at a distance $D = \{7, 9, 11, 13\}$ m from the base edge. These deposits represent heaps of resource that have large volumes and occupy a large area, with $r_D = 0.5$ m and $r_C = 3$ m. For example, a Heap2 scenario contains two deposits with volume of 50 units each.
- **Scatter$N$:** N deposits randomly distributed between $D - 5$ m and $D + 5$ m from the base edge. These deposits are small, with $r_D = 0.1$ m and $r_C = 1$ m and often numerous, containing a small V each. For example, a Scatter25 scenario contains 25 deposits with volume of four units each.

A laden robot returning to the base deposits its load in the unloading area in the form of N_p number of pellets of size 0.1 m^3, where $N_p = L \times (4 \backslash L_{\max})$, set to the closest larger integer. For example, when $L_{\max} = 1$ and $L = 1$, a robot deposits four pellets. When $L_{\max} = 1$ and $L = 0.1$, the robot deposits one pellet, etc. These pellets have the potential to cause congestion in the unloading area if many are deposited at the same time. Deposited pellets disappear from the simulation after a period of *unloading area*

handling time, t_H, representing their use or consumption by a hypothetical unmodelled system of robots or human users. By setting $t_H = 0$ s, the system is able to represent pellets that cause no congestion in the unloading area. Scenarios where $t_H > 0$ s are explored in Section I.8.2

I.3.2 Robots

The simulated MarXbots (Bonani et al., 2010) are circular, differentially steered, robots with a diameter of 0.17 m that in our simulation can reach a maximum speed of 5 cm/s. They are equipped with four colour sensors pointed to the ground, a ring of 24 infra-red proximity sensors used for collision avoidance, a light sensor used for navigation towards the base, a range and bearing module used for localisation of other robots and for communication, wheel mounted sensors utilised for odometry and a ring of eight colour LEDs used for debugging. The maximum range at which the robots can detect objects via the proximity sensors is 0.3 m. Their maximum communication range via the range and bearing module is 5 m. The range and bearing module is based on line of sight, and hence intermittent ranging and communication problems are possible, although they do not have a significant effect in the scenarios we explore here. It is assumed in the simulation that the light sensor can detect the light above the base from anywhere in the experimental arena (as in, e.g., Labella et al., 2004; Gutiérrez et al., 2010). Sensor noise and wheel slippage are not modelled. Consequently, there is no odometry error in our simulation.

The robot control algorithm is inspired by foraging bees and the robots are modelled as finite-state machines (Figure I.3). A robot starts in a random orientation and position on the dance floor as an *observer*, ready to receive recruitment signals. Observers move randomly on the dance floor and avoid traveling into the unloading area. When a recruitment signal is received, the robot becomes a *forager* and navigates towards a deposit location obtained from the recruiter. Alternatively, an observer still on the dance floor can become a *scout* with scouting probability $p(S) = 10^{-3}$ at each time step.

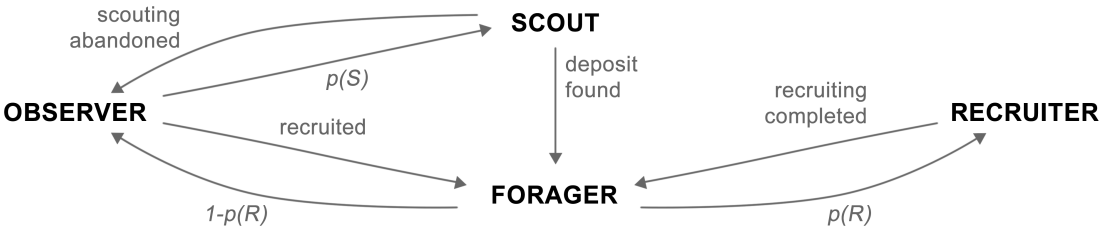


Figure I.3: Finite state machine representation of the robot controller.

Scouts leave the base and use Lévy movement (Reynolds and Rhodes, 2009) to search for a deposit within 20 metres from the base. Any scout that cannot find a deposit within 600 seconds returns to the dance floor and becomes an observer. While outside the base, a robot updates its estimation of the relative position of the base using path integration based on odometry at each time step (e.g., Borenstein, 1998; Lemmens et al., 2008; Gutiérrez et al., 2010). When a scout discovers a deposit, it becomes a forager. All foragers load L units of volume of the resource and determine an estimate of the deposit's energy efficiency (after Seeley, 1994):

$$E_E = \frac{V_i' \times Q_i}{d_i} \tag{I.1}$$

where V_i' is the the volume left in the deposit after the robot's visit, Q_i is the deposit quality and d_i is the odometry-estimated linear distance from the unloading area to the deposit. The robot then returns back to the base utilising phototaxis and keeps track of its relative position to the deposit using odometry.

After a forager unloads its cargo in the unloading area, it moves to the dance floor and becomes a *recruiter* with a recruitment probability $p(R)$:

$$p(R) = \begin{cases} 1.0 & \textit{if } E_E > 0 \\ 0.0 & \textit{else} \end{cases} \tag{I.2}$$

and performs recruitment inspired by bee waggle dancing for $T_R = 120$ seconds, randomly moving across the dance floor and advertising its deposit location to all observers located within communication range d_C. In order to minimise the influence of the particular direction from which it arrived, and thereby give it the chance to influence more observers, the recruiter travels to the middle of the dance floor before it starts recruiting. Similarly, forager bees enter the dance floor from a single direction through a small nest entrance (Seeley and Morse, 1976), allowing potential recruits to interact with any recruiter regardless of its previous foraging location.

The deposit location is communicated to each observer in a one-to-one fashion, where local axes of the robots and their alignment relative to each other are taken into account when conveying the positional information (Gutiérrez et al., 2010). While this technique is sensitive to the accuracy of a robot's sensors (Gutiérrez et al., 2010; Miletitch et al., 2013), it removes the need for a shared reference point. Note that bees, on the other hand, orient their dances relatively to the sun (von Frisch, 1967).

The recruiter always resumes foraging from the same deposit after it completes recruitment. In cases when there is no deposit to return to ($p(R) = 0$), a robot does not recruit or forage again but becomes an observer instead.

If a forager reaches a deposit location but the deposit cannot be found, the robot performs neighbourhood search that lasts 180 simulated seconds and during which it moves randomly in a circular area with a radius of two metres around the expected deposit location. If the search is unsuccessful, the robot returns to the base to become an observer. Any unsuccessful foragers and scouts are opportunistic and start foraging from a deposit if they find one on their way back to the base.

I.4 Foraging performance in static environments

Before exploring scenarios that require plasticity, experiments were performed in Heap1, Heap2, Heap4, Scatter10 and Scatter25 scenarios that lasted $T = 1$ simulated hour and where all deposits had a constant quality $Q = 1$ throughout the simulation. We aimed to understand how the communication range, d_C, and the scouting probability, $p(S)$, of robots affected the total amount of resource collected in these scenarios. Each experiment was repeated 50 times. By varying d_C and $p(S)$ to define three types of robot, we could encourage three types of homogeneous robot swarm: (Figure I.4):

- Solitary robots, where observers left the dance floor almost instantaneously to scout ($p(S) = 10^{-1}$), and robots could never recruit each other ($d_C = 0$ m)
- Short-range recruiters, where observers spent longer on the dance floor ($p(S) = 10^{-3}$), but robots could only recruit observers that were near to them on the dance floor ($d_C = 0.6$ m)
- Long-range recruiters, where robots could recruit any observer on the dance floor ($d_C = 5$ m, $p(S) = 10^{-3}$)

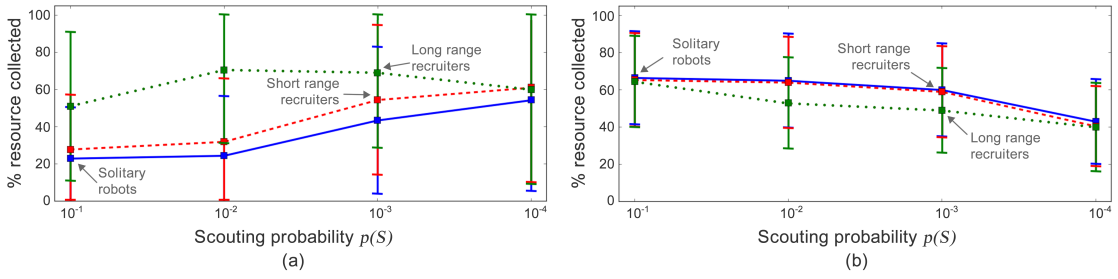


Figure I.4: Foraging performance of 25-robot swarms in the (a) Heap1 and (b) Scatter25 scenarios using various $p(S)$ values and $d_C = 0$ m (solid line), $d_C = 0.6$ m (dashed line) and $d_C = 5$ m (dotted line). Each point represents mean percentage of available resource that was collected in a given scenario, collated over 50 one-hour-long runs for each of the deposit distances in the set $D = \{7, 9, 11, 13\}$ m. The whiskers represent 95% confidence intervals.

In the most extreme Heap scenario, Heap1, with only a single large deposit, using large d_C and small $p(S)$ was the most beneficial as it allowed for mass recruitment once the deposit was discovered and foragers returned to the base to convey the deposit location to any unsuccessful scouts, making long-range recruiters the most suitable option (Figure I.4a). In the other extreme scenario, Scatter25, where there were 25 small deposits, solitary foraging led to the best results (Figure I.4b). In this case, recruitment damaged performance because of environmental interference between robots, where the robots altered each other's foraging environment by depleting low-volume deposits and where recruitment thus led to a lot of wasteful foraging trips (see Pitonakova et al., 2014, for discussion of robot-robot interference during foraging). Short-range recruiters could achieve a balance between exploration and exploitation, which made them more robust across scenarios, although they never achieved performance better than that of the other two swarm types (see the dashed line in Figure I.4).

The results shown in Figure I.4 were obtained using swarms of 25 robots. Similar relationships between d_C and $p(S)$ were found for swarms of 15 and 35 robots. The findings also held for less extreme environments such as Heap2, Heap4 or Scatter10, although results varied between runs more in these intermediate scenarios.

I.5 Heterogeneous deposit quality

Having established that recruitment is most beneficial in Heap scenarios, where deposits are large but hard to find, we now explore what type of information transfer would allow the robots to collectively choose a deposit of a better energy efficiency. We examined the following scenarios using deposit distances $D = \{7, 9, 11, 13\}$ m:

- **Heap2A:** two deposits with volume $V = 50$ each and with qualities $Q_1 = 0.5$, $Q_2 = 1.5$
- **Heap2B:** two deposits with $V = 50$ each and with qualities $Q_1 = 0.1$, $Q_2 = 1.9$
- **Heap4A:** four deposits with $V = 25$ each and with qualities $Q_{1-3} = 0.5$, $Q_4 = 2.5$
- **Heap4B:** four deposits with $V = 25$ each and with qualities $Q_1 = 0.1$, $Q_2 = 0.5$, $Q_3 = 1.5$, $Q_4 = 1.9$

The maximum volume $L_{\max}$ loaded by a robot per foraging trip was decreased from 1.0 to 0.25 in all experiments reported below. The robots still collected $L_{\max} \times Q$ units of net resource value per foraging trip and deposited a maximum of four pellets in the unloading area. Decreasing $L_{\max}$ causes swarms to carry out more foraging trips during a simulation (see online supplementary material, Figure S1), generating a larger data set

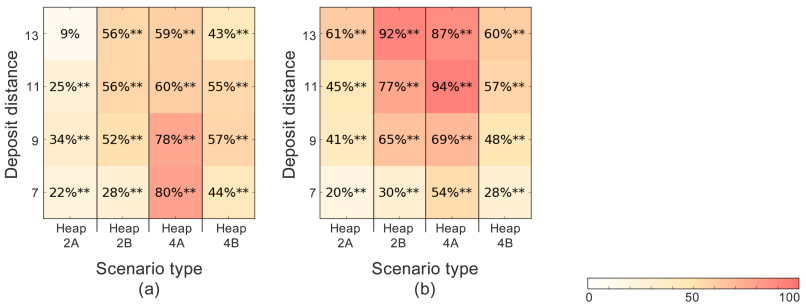


Figure I.5: Difference in the amount of resource collected compared to control (non-switching) swarms using 25 beggers with (a) short-range, (b) long-range signals during one simulated hour. The measured values represent averages from 50 independent runs. Statistically significant differences are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

which is necessary in order to analyse foraging dynamics when the environment changes over time.

The swarms used in the previous experiments were unable to discriminate between deposits based on energy efficiency as robots always followed the first recruiter that they met and remained foraging from a deposit until it was fully depleted. On average, such swarms thus distributed themselves to forage from all deposits equally. They were used as control swarms in the following experiments.

In order to allow robots to preferentially concentrate their foraging effort on deposits with higher energy efficiency, we created swarms of ***beggers***, where all robot states and state transitions remained the same, but where recruiters asked each other about the energy efficiency of advertised deposits. If another robot's deposit had higher E_E, a recruiter switched to advertise this deposit and then foraged from it. This strategy was inspired by the begging behaviour of bees described in Section I.2.1[1]. As before, we tested beggers with both short-range (0.6 m) and long-range (5 m) communication signals.

Swarms of short-range beggers were able to identify and concentrate on better deposits within the first 20 minutes and could thus collect more resource than the control swarms (Figure I.5a). Long-range beggers switched to better deposits faster, making them more efficient at the task, especially for larger deposit distances (Figure I.5b). The size of the difference between deposit qualities did not affect the time it took to switch to a better deposit given a particular number of deposits. However, performance of both swarms was better in the Heap2B scenarios compared to Heap2A, as the difference between deposit qualities was larger in Heap2B and it was thus more advantageous to forage from the better deposit. Similarly, when robots could concentrate on a deposit with

[1]The begging behaviour of our robots is simplified compared to that of bees. Bees integrate information received from multiple nest mates and their foraging preferences can be affected for a number of days (De Marco and Farina, 2001; Farina et al., 2005).

quality 2.5 in Heap4A, they performed better than in Heap4B, where the best deposit had quality of only 1.9.

I.6 The ability of robots to repeatedly switch between deposits

In biological settings such as nectar foraging, the quality of foraging sites changes over time. In robotic applications, a preference for where to forage may also need to change due to the nature of the task, orders from humans, etc. It is thus important not only to be able to collectively select a better foraging site, but to be able to do so repeatedly, as the environment changes. Therefore, in dynamic environments, the ability to achieve a balance between exploitation and exploration becomes important.

To test the ability of our swarms to preferentially forage from better deposits in dynamic environments over a long period of time, we ran simulations over a number of hours and changed deposit qualities (but not their locations) at the end of each *deposit quality*

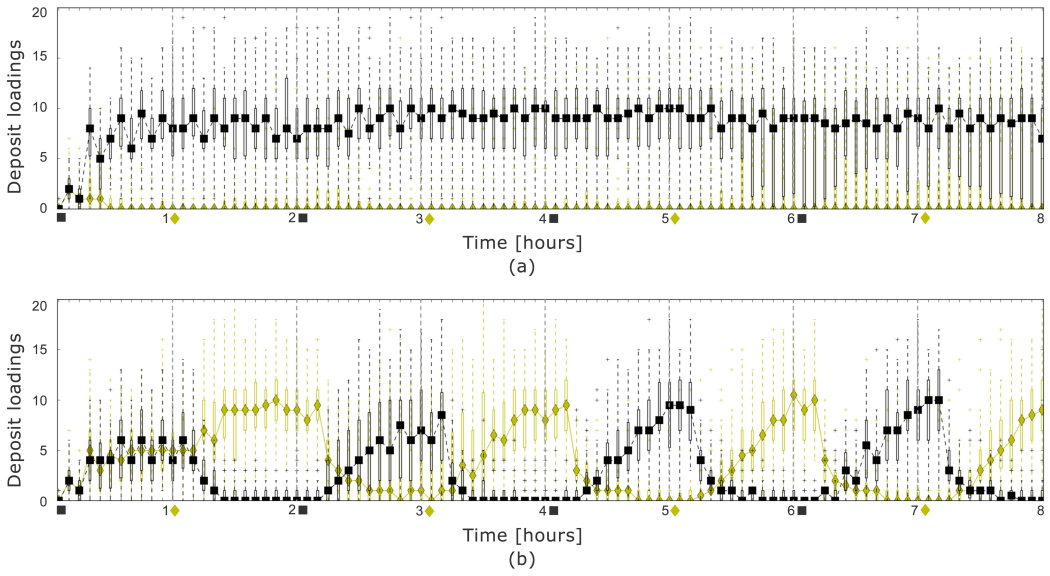


Figure I.6: Median number of loadings from two deposits (square and diamond symbols) during the first eight hours of simulation using (a) 25 short-range beggers and (b) 25 short-range checkers in the Heap2B scenario with $D = 9$ m, $T_Q = 1$ h. Deposit qualities were exchanged every hour during the experiment. The symbol of the deposit with higher quality is shown along the time axis at the beginning of each hour. Each data point represents a median value for a particular time interval and is based on a set of results collected from 50 independent runs. A data point is surrounded by a box, representing the interquartile range or "middle fifty" of the result set, and whiskers representing the "middle 97", with outliers outside this range shown as plus signs.

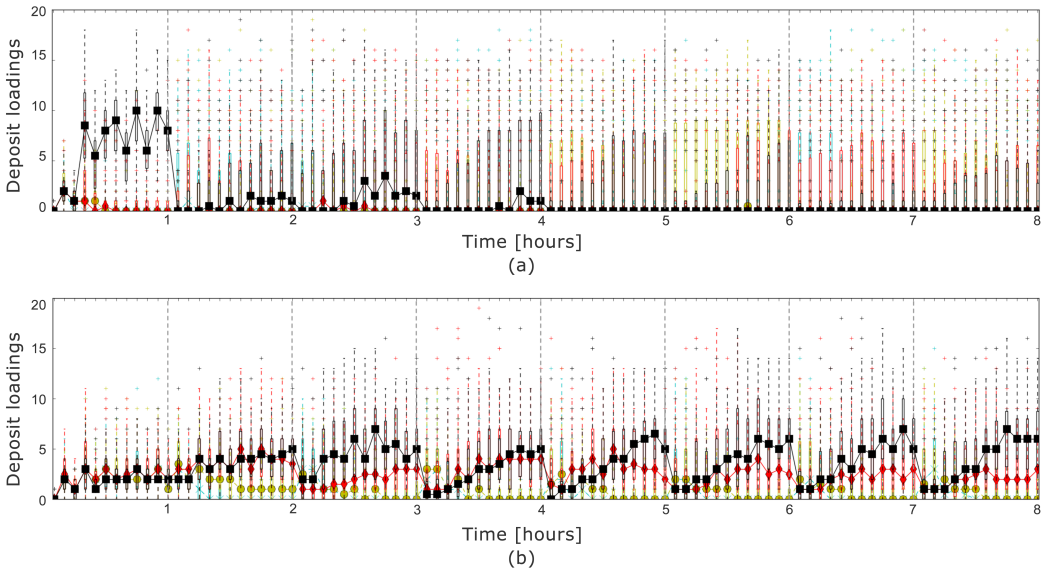


Figure I.7: Median number of loadings from four deposits during the first eight hours of simulation using (a) 25 short-range beggers and (b) 25 short-range checkers in the Heap4B scenario with $D = 9$ m, $T_Q = 1$ h, based on 50 independent runs. Deposit qualities were assigned randomly every hour during the experiment. Therefore, instead of identifying loadings from a particular deposit in a particular location, like in Figure I.6, each symbol identifies loadings from a deposit of a particular quality in a given time interval: $Q = 1.9$ (squares), $Q = 1.5$ (diamonds), $Q = 0.5$ (circles), $Q = 0.1$ (crosses).

change interval of length T_Q. When the change was made, deposits were assigned new qualities from a quality set given for a particular scenario and their volumes were replenished, so that the amount of available resource was the same at the beginning of each quality change interval.

In this task, strong commitment to a single best deposit could be a serious problem, as it might cause the swarm to loose track of other deposits and of new places to forage from when the quality change occurs. This was the case in all Heap2 scenarios, where beggers locked into foraging from a single deposit (Figure I.6a), an effect that was more pronounced for long-range beggers, and that held for both short ($T_Q = 1$ h) and long ($T_Q = 2$ h) quality change intervals.

On the other hand, in all Heap4 scenarios, where deposits had only half of the volume compared to the Heap2 scenarios, and the differences between their energy efficiencies were thus smaller, the resulting foraging pattern depended on the communication range of robots and the length of the quality change interval used (T_Q). Since beggers usually rapidly recruited each other to the best deposit found, large groups of robots quickly exploited a single foraging site until its energy efficiency fell below that of alternative deposits, at which point the robots rapidly switched to a different deposit. This repeated every time a certain volume of a deposit that was currently being exploited by

a majority of robots was depleted, preventing the swarms from preferentially foraging from deposits with superior quality and, on average, to forage from all deposits with an equal probability (Figure I.7a). Unlike in Heap2, some members of the swarm always foraged from inferior deposits and the swarm thus retained the memory of all deposits in the environment. However, the inability to forage from deposits of superior quality for sufficiently long caused short-range beggers when $T_Q = 1$ h and long-range beggers when $T_Q = 1$ h and 2 h to collect less resource that the control swarms (Figures I.8a,b and I.9a,b). On the other hand, when $T_Q = 2$ h, deposit quality remained stable long enough, allowing short-range beggers to eventually choose the best deposit after a period of indecisiveness, and more importantly to repeat this pattern during each quality

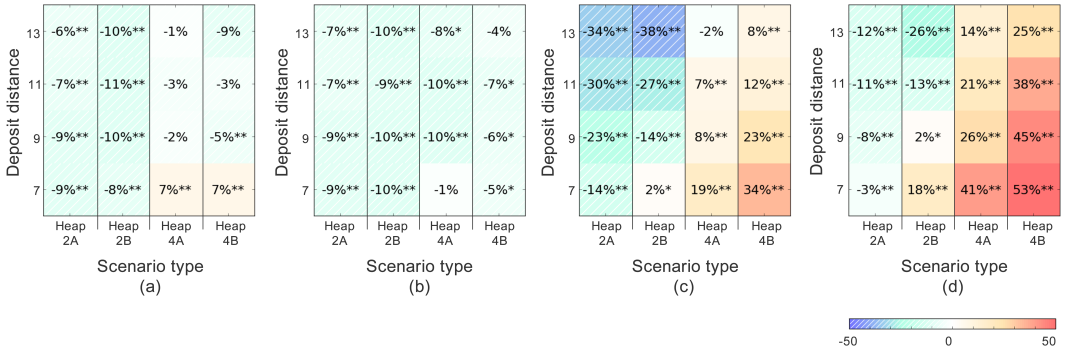


Figure I.8: Difference in the amount of resource collected compared to control (non-switching) swarms using (a) 25 short-range beggers, (b) 25 long-range beggers, (c) 25 short-range checkers (d) 25 long-range checkers during 1 h $< T <$ 13 h of 50 independent runs, using $T_Q = 1$ h. Statistically significant differences are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

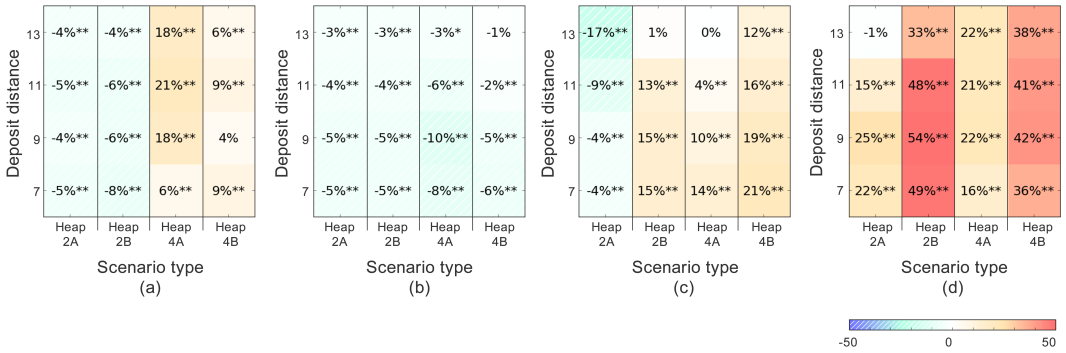


Figure I.9: Difference in the amount of resource collected compared to control (non-switching) swarms using (a) 25 short-range beggers, (b) 25 long-range beggers, (c) 25 short-range checkers (d) 25 long-range checkers during 2 h $< T <$ 26 h of 50 independent runs, using $T_Q = 2$ h. Statistically significant improvements are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

change interval. This led to foraging performance better than that of the control swarms in both Heap4A and Heap4B scenarios when deposits were near the base (Figure I.9a).

These results suggested that a swarm behaviour where the spread of social information is regulated could potentially lead to more effective plastic reconfiguration and consequently more sustainable preferential foraging. To test this hypothesis, we introduced a modified behaviour that explicitly regulated the spread of social information. A swarm of ***checkers*** was implemented, where the robots did not compare deposit energy efficiency directly with other recruiters, as it was the case with beggers. However, instead of having a binary recruitment probability, $p(R)$ (Equation I.2), and a fixed recruitment time, T_R, checkers calculated these variables based on how the energy efficiency of a deposit had changed since the last time they had visited it:

$$\delta = \min\left(1.0, \frac{E_E}{E'_E}\delta'\right) \tag{I.3}$$

where E'_E and δ' were the energy efficiency of the deposit and δ measured during the last loading event,

$$p(R) = \delta \tag{I.4}$$

$$T_R = \delta \times 120s \tag{I.5}$$

Because their recruitment probability, and consequently their probability of returning to a deposit, decreased gradually as the deposit became depleted, checkers abandoned a foraging site one by one over time, allowing them to be recruited by other robots or to go scouting. By contrast, both the control swarms and beggers remained foraging from the same deposit until it became completely depleted or until, in the case of beggers, the robots received social information about a better deposit. Perhaps more importantly, the fact that checkers evaluated the change in energy efficiency of a deposit each time they visited it, enabled them to immediately abandon a deposit if its quality suddenly decreased, without having to rely on receiving knowledge from other robots about better deposits, as the beggers did.

The checker behavioural strategy was also inspired by bees, in particular by their food patch abandonment behaviour (Seeley, 1994; Biesmeijer and De Vries, 2001). An agent-based model, where recruitment time was influenced by quality of an advertised site but where agents, similarly to our beggers, could also switch to another site after recruitment was explored by (Valentini et al., 2014).

The checker strategy allowed the swarms to repeatedly switch to a better deposit in both Heap2 (Figure I.6b) and Heap4 (Figure I.7b) scenarios, unlike the beggers that either foraged from one deposit throughout the whole run or were often unable to make a clear collective decision. The checker swarms were able to concentrate on exploiting better deposits faster when deposit distance was smaller (see online supplementary material,

Figures S2 and S3), as deposit visits, as well as recruitment, occurred more frequently (for example in Heap2B, the swarms switched to a better deposit after about 25 minutes when $D = 7$ m and after about 35 minutes when $D = 13$ m). However, note that checker swarms were unable to concentrate on the better deposit before the first quality change occurred, as a majority of robots had to abandon the deposit from which they initially foraged before recruitment to a better foraging site could take place. Such a significant abandonment of a deposit could only occur if the deposit was nearly depleted, which did not occur unless the whole swarm foraged from it, or when deposit qualities were changed suddenly by the simulation engine.

Nevertheless, the long-term behaviour of the checkers was much more desirable than that of the beggers and could significantly outperform the control swarms, especially when four deposits were placed in the environment, when the deposit quality change interval was long, or when long-range recruitment signals were used (Figures I.8c,d and I.9c,d). However, in some scenarios, the checkers collected less resource than the control swarms. This was the case in all Heap2 scenarios when $T_Q = 1$ h for both short- and long-range checkers and in the Heap2A scenario when $T_Q = 2$ h for short-range checkers. The poor performance was caused by the fact that the checkers gradually abandoned a deposit of a worse quality after deposit qualities were changed, which led to a decrease in foraging activity. The robots that stopped foraging waited on the dance floor to be recruited or left the base to become scouts, creating a period of time when the swarm could not do any foraging work. Even though the robots eventually did forage from a deposit of a better quality, they often did not have enough time to make up for the loss of foraging time. The period of low foraging activity was especially long for short-range checkers and when the deposit distance was large. For example, it took the swarms up to 35 minutes to start foraging from the better deposit in the Heap2B scenario when $D = 13$ m and $T_Q = 1$ h (see online supplementary material, Figure S2), leading to 38% decrease in the total amount of resource collected compared to the control swarm (Figure I.8c).

I.7 Analysing the value of information

It is clear that foraging performance in dynamic environments is closely related to how information is obtained and exchanged between robots. While sudden bursts of information transfer can lead to over commitment to a single deposit, which reduces the chance of acquiring information on alternative deposit sites, a mix of scouting and more controlled information transfer can lead to plasticity. In order to characterise behavioural strategies that are beneficial, it would be useful to have a measure that could quantify the effect of communication.

It is possible to mathematically model swarms using differential equations and then use the model to test simulation parameters and predict the average result (Lerman et al., 2006; Liu and Winfield, 2010). It has also been shown that decision-making swarms can be modelled as dynamical systems that exhibit Hopf bifurcations and limit cycles (Pais et al., 2012). Finally, information theoretic measures, such as transfer entropy and local data storage, can be used to show how communication affects the state of an agent based on the states of the other agents it shares information with (Miller et al., 2014).

While mathematical models are often tractable and can generalise well (Martinoli et al., 2004), they are also often complex and difficult to apply in real-life scenarios (McFarland and Spier, 1997) where many options exist and where physical interactions substantially influence the resulting swarm behaviour. Inspired by information theoretic approaches, we seek to establish a simple measure, the information value I, that works directly from experimental data and represents a quantifiable effect of information transfer between the environment and the swarm and within the swarm itself. This measure should reflect the flow of new information though the system, should be positive when recruitment leads to exploitation of relatively good deposits and should be negative when recruitment causes the system to lower its foraging performance.

The information value I_r of a robot is defined as

$$I_r = Q_{\text{new}} - Q_{\text{old}} \tag{I.6}$$

where Q_{new} is the quality of a new deposit the robot finds out about and Q_{old} is the quality of a deposit it previously foraged from. Note that $Q_{\text{old}} = 0$ when a robot has not foraged before. I_r is thus equal to Q_{new} when a scout discovers a new deposit or when a robot that does not posses any information is recruited. In cases when a robot currently foraging from a deposit switches to another, I_r can be either positive or negative. For example, if a robot previously foraged from a deposit with $Q_i = 1.0$ and was recruited to a new deposit with $Q_i = 0.5$, $I_r = -0.5$. Finally, $I_r = 0$ for all robots that do not receive any information in a given time step.

We compare deposit qualities, rather than energy efficiencies, i.e., we do not take the volume left in a deposit or its distance from the base into account. The information value thus only captures utility obtained from a single foraging trip that the robot is guaranteed to make, rather than that of possible future trips. Furthermore, since deposits in our Heap scenarios have the same distance from the base, comparing their qualities is sufficient to identify which one is more profitable.

We obtain the swarm information value I normalised per robot as:

$$I = \sum_{r=1}^{N_R} I_r \times \frac{1}{N_R} \tag{I.7}$$

where N_R is the total number of robots. A time series of the swarm (Figure I.10) is obtained by sampling the average value of I over short time intervals (in our case 300-second intervals). When a swarm working in a dynamic environment is able to repeatedly switch its attention to the current best deposit, the shape of the time series repeats for each quality change interval and has three distinguishable information value regions:

1. **A negative region** that is caused by recruitment to the previously exploited deposit(s) that have become worse since the change in deposit quality.
2. **A flat region** where new and old information coexists in the system. We refer to the negative region and the flat region together as a non-positive region.
3. **A positive region** that represents recruitment to the newly discovered better-quality deposit(s).

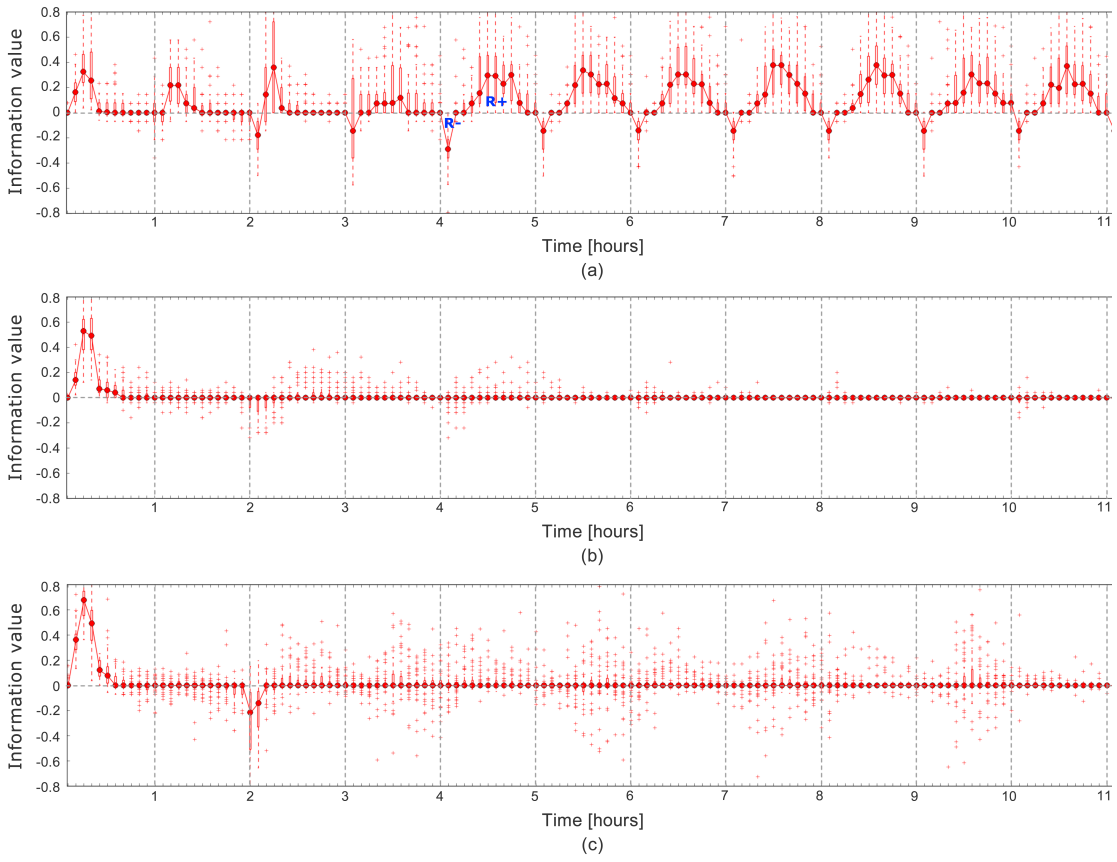


Figure I.10: An information value time series based on 50 independent runs with 25 robots for (a) switching swarm: long-range checkers in Heap2B, $D = 9$ m, $T_Q = 1$ h (b) locked swarm: long-range beggers in Heap2A, $D = 9$ m, $T_Q = 2$ h (c) indecisive swarm: long-range beggers in Heap4B, $D = 9$ m, $T_Q = 2$ h. Negative regions (R-) and positive regions (R+) are shown for the switching swarm.

The size (i.e., the area between the curve and the abscissa) of negative regions represents a lag between changes in the environment and changes in the swarm's collective knowledge. This region is usually larger for long-range recruiters and for shorter deposit distances, i.e., when recruitment is stronger (see online supplementary material, Figures S4–S6). Stronger recruitment tends to result in a larger number of robots to be foraging from the better deposit at the end of a change interval. Immediately after a quality change, there is thus a higher probability of an observer being recruited by a robot that is signalling out-of-date information.

The size of positive regions is influenced by the number of observers that a recruiter holding new information can reach, i.e., by the speed of information transfer through the swarm. It is also larger for long-range recruiters and shorter deposit distances.

The length of the non-positive regions is a combination of the positive impact of information speed and the negative impact of commitment to old information.

When robots deplete and abandon the better deposit(s) before the quality change occurs, as was the case for checkers in Heap2 scenarios when $T_Q = 2$ h, there is no recruitment to a low-quality deposit after the quality change. Therefore, there are no negative regions of I and non-positive regions become very short. On the other hand, when the whole swarm commits to a single deposit and is unable to abandon it throughout the experiment, as was usually the case with long-range beggers, only a single large positive region exists at the beginning of the simulation, followed by zero information value throughout the rest of the run. Using information value, we can thus distinguish three work modes of foraging swarms (Figure I.10):

1. ***Switching*** swarms that alternate between non-positive and positive regions and sometimes also show negative regions. These swarms are able to concentrate on deposit(s) of better quality in each quality change interval.

2. ***Locked*** swarms that show a large positive region at the beginning of the simulation, but are unable to alter their chosen foraging site for the rest of the run and thus show zero communication value thereafter.

3. ***Indecisive*** swarms that also have a large positive region at the beginning, but are able to gain new information through scouting or because recruitment does not affect as many robots as in locked swarms. In contrast with switching swarms, rapid information transfer in indecisive swarms prevents the collective from settling on one solution, leading to oscillations around $I = 0$. When multiple runs are considered, time series of I has medians of zero and many outliers.

When runs that last several hours and have multiple quality change intervals are considered, locked and indecisive swarms always collect less resource than the control swarms

that forage from all deposits with similar proportions. The relative performance of switching swarms depends on a number of factors. As we showed in the previous sections, switching to better deposits can sometimes occur too late or the total amount of loadings can be insufficient, which leads to worse performance. This was for example the case for checkers in Heap2 scenarios when $T_Q = 1$ h.

In the following sections, we test our communication strategies further under various experimental perturbations and use the information value measure, I, to analyse the results. We show that the switching work mode is more common when information transfer is slower and that measuring I can thus help robot designers to select behavioural parameters that lead to effective plastic self-organisation.

I.8 Varying experimental parameters

It is important to understand how the chosen experimental parameters affect the performance and work modes of our swarms. In this section we explore the influence of the shape of the dance floor, of the tendency for collected pellets to accumulate and interfere with robot movement, and of the size of the robot swarms. We evaluate how varying these aspects of the foraging scenario affects the behaviour of all four types of swarms (i.e., short-range and long-range beggers, and short-range and long-range checkers) compared to performance under normal conditions (i.e., circular dance floor, no pellet accumulation, and 25-robot swarms – see Sections I.5 and I.6). We also identify cases in which variation to these aspects of the scenario causes a swarm to transition from one work mode to another (e.g., from switching to indecisive). See Figure I.11 for a summary of all experiments and the work modes exhibited by the swarms.

I.8.1 Restricted dance floor shape

In the previous experiments, a returning forager traveled to the middle of the base before it started recruiting in order to be able to interact with any observer, regardless of direction from which it returned to the base. However, the constraints of a particular foraging scenario might require a different approach. For example, we might want to create a recharging area in a part of the recruitment area or otherwise change the shape of the base and thus indirectly alter the way in which robots meet each other and communicate. Information flow throughout a swarm will tend to be influenced by such physical logistics, potentially changing the ability of the swarm to maintain plasticity and thereby influencing performance levels.

In the following experiment, we designated an inaccessible circular area in the middle of the base, making the dance floor doughnut-shaped and 35 cm thick. While the robots could still see other robots across the inaccessible central area, and could thus

		Short-range beggers	Long-range beggers	Short-range checkers	Long-range checkers
Normal conditions	**Heap2, TQ = 1h**	Locked	Locked	Switching	Switching
	Heap2, TQ = 2h				
	Heap4, TQ = 1h	Indecisive	Indecisive		
	Heap4, TQ = 2h	Switching			

		Short-range beggers	Long-range beggers	Short-range checkers	Long-range checkers
Restricted dance floor shape	**Heap2, TQ = 1h**	**Delayed switching**	Locked	Switching	Switching
	Heap2, TQ = 2h				
	Heap4, TQ = 1h		Indecisive		
	Heap4, TQ = 2h	Switching			

		Short-range beggers	Long-range beggers	Short-range checkers	Long-range checkers
Increased pellet accumulation	**Heap2, TQ = 1h**	Locked	Locked	Switching	Switching
	Heap2, TQ = 2h				
	Heap4, TQ = 1h	Indecisive	Indecisive	**Indecisive**	**Indecisive**
	Heap4, TQ = 2h	**Indecisive**			

		Short-range beggers	Long-range beggers	Short-range checkers	Long-range checkers
Increased swarm size	**Heap2, TQ = 1h**	**Delayed switching**	**Delayed switching**	Switching	**Delayed switching**
	Heap2, TQ = 2h	**Switching**			Switching
	Heap4, TQ = 1h		**Switching**		
	Heap4, TQ = 2h	Switching			

Figure I.11: Work modes exhibited by the swarms under various experimental setups. Work modes that differ from those exhibited under normal conditions (see text) are shown in bold.

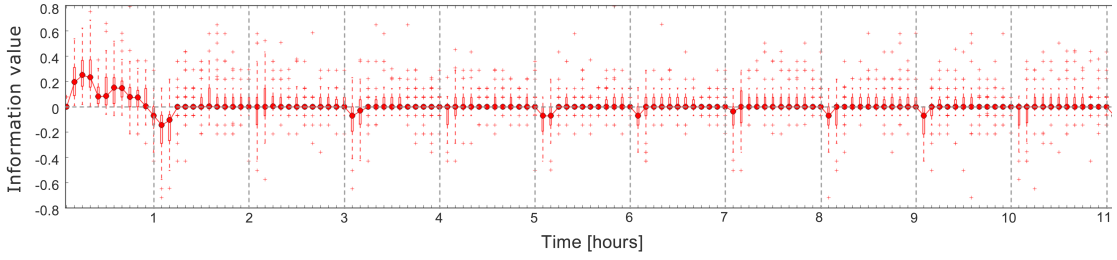


Figure I.12: An information value time series based on 50 independent runs for a delayed switching swarm of 25 short-range beggers in Heap2B, $D = 9$ m, $T_Q = 1$ h with restricted dance floor shape.

communicate with them if their communication range allowed for it, they could not move through the central restricted area. Consequently, while undertaking a random walk within the dance floor, a robot tended to remain relatively close to the point at which it first entered the base. The restricted shape of the dance floor also resulted in higher congestion and constrained movement of the robots within and immediately

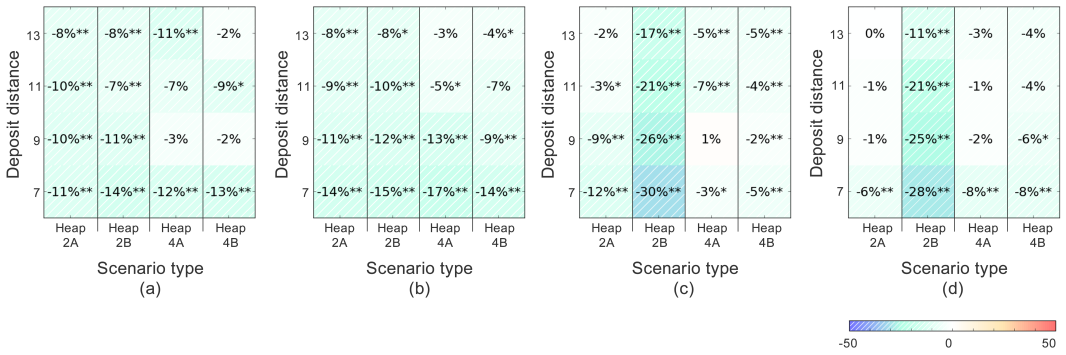


Figure I.13: Difference in the amount of resource collected using an annular dance floor compared to the same communication strategy operating over a full dance floor with (a) 25 short-range beggers (b) 25 long-range beggers, (c) 25 short-range checkers, (d) 25 long-range checkers during 1 h $< T <$ 13 h of 50 independent runs, using $T_Q = 1$ h. Statistically significant differences are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

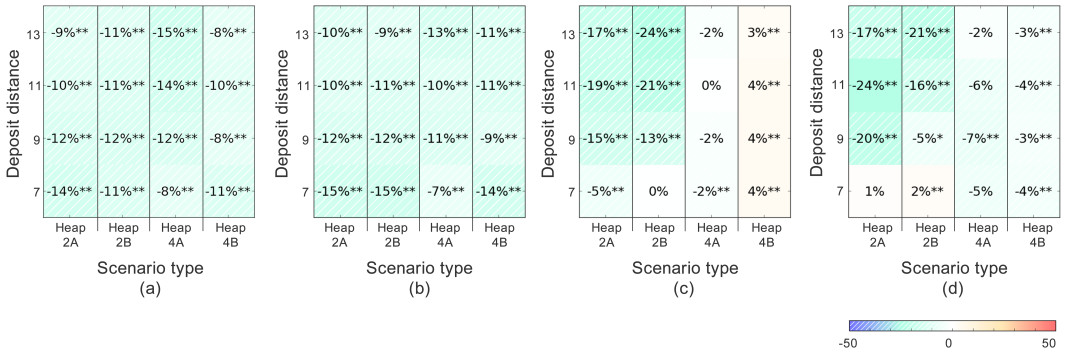


Figure I.14: Difference in the amount of resource collected using an annular dance floor compared to the same communication strategy operating over a full dance floor with (a) 25 short-range beggers (b) 25 long-range beggers, (c) 25 short-range checkers, (d) 25 long-range checkers during 2 h $< T <$26 h of 50 independent runs, using $T_Q = 2$ h. Statistically significant differences are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

around the dance floor. The congestion was more severe when long-range recruitment was used or when deposits were closer to the base, i.e., when the number of foragers returning to the base at the same time was high. Additionally, the annular shape of the dance floor restricted communication within swarms with short-range recruitment, as it made interactions between robots that foraged from different areas of the environment less probable and prevented information from spreading as easily throughout the swarm.

Although most swarms did not change their work modes, information value analysis showed that restricting dance floor shape decreased the rate at which new information

about the best deposit(s) spread through the swarm (see online supplementary material, Figures S7 and S8). For example, the total size of positive regions (i.e., sum of all $I > 0$) was 40% smaller for short-range checkers in Heap2s ($D = 9$ m and $T_Q = 1$ h) compared to normal conditions. The size of positive regions was less affected for long-range checkers (only 28% smaller), but higher congestion caused the positive regions to appear much later than under normal conditions (usually after 35 instead of 25 minutes).

Short-range beggers were the only swarm that exhibited different work modes compared to the normal conditions (Figure I.11). The decreased probability of interactions between recruiters and observers on the dance floor caused the swarms to escape the locked mode they experienced with a full dance floor in Heap2 scenarios, allowing them to preferentially forage from better deposits. However, clear positive regions of information value were absent while negative regions remained, meaning that the information about new better deposits spread too slowly to be of use. We term this a *delayed switching* work mode (Figure I.12). The swarms also changed their work mode from indecisive to delayed switching in Heap4 scenarios with $T_Q = 1$ h, as constrained information transfer prevented the confusion that had previously resulted from strong competition between deposits with equivalent energy efficiencies. Despite the fact that, when in the delayed switching mode, short-range beggers were able to alternate between deposits as the deposit quality changed, the foraging performance of the swarms deteriorated compared to normal conditions (Figure I.13a and I.14a). First, the total number of foraging trips was smaller due to congestion caused by a smaller dance floor. Second, the delayed switching mode led to a similar number of loadings from deposits of higher and lower quality when the whole run was considered, making the proportion of loadings from the better deposit(s) similar to when the swarms were in the locked or indecisive modes (see online supplementary material, Figures S9 and S10).

The impact of congestion caused by a restrictive dance floor was most clearly observed for long-range beggers. While a recruiter's signal could reach any observer on the dance floor regardless of their relative positions and the work mode of the swarms thus remained locked, foraging performance still deteriorated as the robots spent significantly more time avoiding each other (Figures I.13b and I.14b).

While the restricted movement of recruiters affected both the short- and long-range checkers negatively in almost all scenarios (Figures I.13c,d and I.14c,d), it had a surprisingly positive effect on short-range checkers in Heap4B scenarios when $T_Q = 2$ h (Figure I.14c). In this case, the impaired ability of robots to recruit each other caused the swarms to forage from the best two deposits in parallel and thus to collect a total amount of resource that was higher than under normal conditions.

I.8.2 Pellet accumulation

The material collected by robots in the previous experiments disappeared instantly when it was placed in the unloading bay, i.e., the unloading area handling time $t_H = 0$ s. However, in real world applications, material would have to be stored somewhere for later use by humans or other robots. It is therefore reasonable to assume that it would accumulate and thus affect the swarm's ability to collect more. To test performance of our swarms under such conditions, we set $t_H = \{10, 20\}$ s, i.e., pellets dropped by robots could accumulate in the unloading area and create congestion. Since the congestion prevented robots from foraging with maximum efficiency, and from reaching the dance floor in order to communicate, swarms always collected less resources than under normal conditions.

When pellets accumulated, both the negative and positive regions of information value were more flat, while the non-positive regions usually became longer, indicating that limited access to and from the base prevented information transfer. This effect was observed more strongly for long-range recruiters and the higher value of t_H (see online supplementary material, Figures S11 and S12), i.e., when more foragers returned to the base at the same time and when pellets remained in the unloading area for a longer period of time. For example, when $t_H = 20$ s, the total size of positive regions (i.e. sum of all $I > 0$) in Heap2s ($D = 9$ m and $T_Q = 1$ h) was 67% smaller for short-range checkers and 75% smaller for long-range checkers compared to the normal conditions. The length of non-positive regions decreased by 25% for short-range checkers and increased by 60% for long-range checkers.

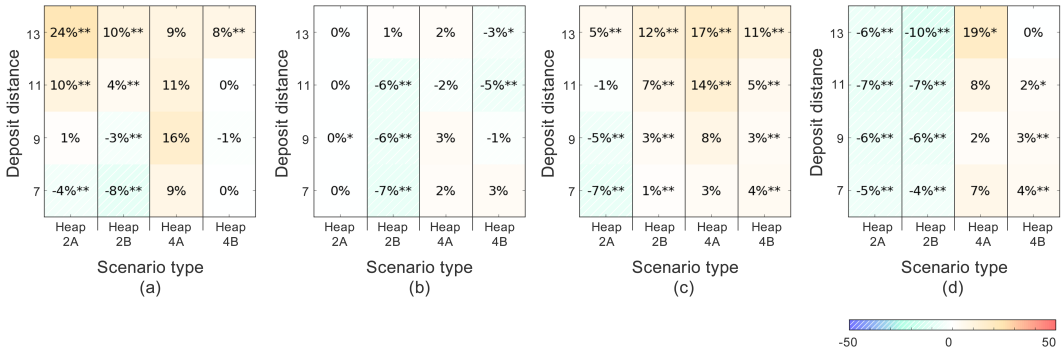


Figure I.15: Improvement of resource collected using 25 long-range checkers compared to 25 short-range checkers during 50 independent runs with (a) unloading area handling time $t_H = 10$ s, $T_Q = 1$ h and 1 h $< T <$ 13 h, (b) $t_H = 20$ s, $T_Q = 1$ h and 1 h $< T <$ 13 h, c) $t_H = 10$ s, $T_Q = 2$ h and 2 h $< T <$ 26 h, d) $t_H = 20$ s, $T_Q = 2$ h and 2 h $< T <$ 26 h. Statistically significant improvements are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

Since the rate at which the robots communicated was affected, the ability of swarms to switch between deposits was impaired. This caused short-range beggers, short-range checkers and long-range checkers to become indecisive in Heap4 scenarios (Figure I.11). There were no significant changes in work modes observed for long-range beggers, as they already exhibited the locked (in Heap2) and indecisive (in Heap4) work modes under normal conditions.

Pellet accumulation had also an interesting effect on the advantage of long-range over short-range recruitment. For example, under normal conditions, long-range checkers always outperformed short-range checkers in terms of the total amount of resource collected due to their ability to recruit to the best deposit faster. However, when pellets accumulated, this faster recruitment resulted in stronger congestion and in subsequent worse performance compared to short-range checkers in some scenarios (Figure I.15). This was especially true in the Heap2 scenarios when the unloading area handling time $t_H = 20$ s and when the swarms were given a long time to forage from the selected deposit ($T_Q = 2$ h) (Figure I.15d).

I.8.3 Increased swarm size

All previous experiments were performed with swarms of 25 robots. Here we simulate swarms of 45 and 65 robots.

While short- and long-range beggers previously exhibited the locked work mode in Heap2 scenarios, higher scouting success of the swarm, caused by a higher number of robots that were scouting at the beginning of a simulation run, led to a more even distribution of robots between the deposits. This allowed larger swarms to preferentially forage from better deposits and exhibit the delayed switching work mode, or the switching work mode in the case of short-range beggers when $T_Q = 2$ h or when $N_R = 65$ (see Figure I.11 and online supplementary material, Figure S13). However, the significant delays in recruitment to a better deposit usually caused a stronger disadvantage for the beggers over the control swarms compared to when the number of robots $N_R = 25$ (compare Figure I.16a, b with Figure I.8a,b and Figure I.17b with Figure I.9b). In Heap4 scenarios, where the beggers were previously indecisive, the swarms were able to achieve the switching mode regardless of the value of T_Q (Figure I.11). The switching was effective enough to allow both short- and long-range beggers to outperform the control swarms when $N_R = 45$. For example, short-range beggers collected 17% more resources than the control swarms of the same size in Heap4A when $D = 7$ m and $N_R = 45$. However, their advantage largely disappeared when $N_R = 65$ (Figures I.16a and I.17a) due to rapid exploitation of the better foraging sites and consequent ineffective foraging from other deposits, as well as due to increased congestion in the base. The disadvantage of 65-robot swarms was more pronounced for long-range beggers (Figures I.16b and

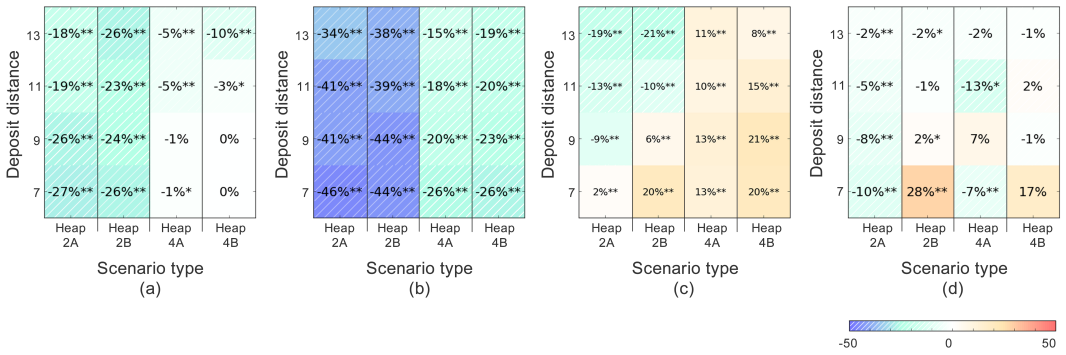


Figure I.16: Improvement of resource collected compared to control (non-switching) swarms of 65 robots using (a) 65 short-range beggers, (b) 65 long-range beggers, (c) 65 short-range checkers (d) 65 long-range checkers during 1 h $< T <$ 13 h of 50 independent runs, $T_Q = 1$ h. Statistically significant improvements are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

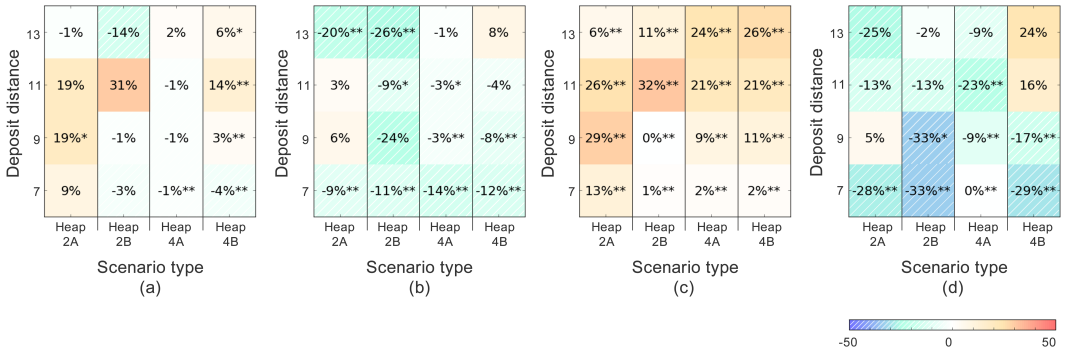


Figure I.17: Improvement of resource collected compared to control (non-switching) swarms of 65 robots using (a) 65 short-range beggers, (b) 65 long-range beggers, (c) 65 short-range checkers (d) 65 long-range checkers during 2 h $< T <$ 26 h of 50 independent runs, $T_Q = 2$ h. Statistically significant improvements are indicated with asterisks (Wilcoxon signed-rank test, $** = p < 0.01$, $* = p < 0.05$).

I.17b) that shared information with each other faster and thus exploited deposits more rapidly.

Checkers generally retained their ability to switch between deposits when the swarm size increased. The information value of the swarms, normalised per robot, did not change significantly, although the positive regions at the beginning of simulation runs were more prominent compared to when 25 robots were used (see online supplementary material, Figure S14). This suggests that information spread faster at the beginning of a run, causing the swarms to be able to exhibit periodic switching behaviour sooner. As with beggers, checkers performed better than the control swarms when $N_R = 45$.

Long-range checkers lost their advantage when $N_R = 65$ for similar reasons to the beggers (Figures I.16d and I.17d). It was also observed that 65-robot swarms of long-range checkers exhibited the delayed switching work mode instead of the switching work mode in Heap2 scenarios when $T_Q = 1$ h as a result of very significant commitment to a single deposit in each quality change interval and consequent insufficient scouting. Furthermore, there was a large variance in the total amount of collected resource across multiple runs when long-range checkers were used and when $N_R = 65$, $T_Q = 2$ h. On the other hand, large swarms of short-range checkers, where information could not spread as quickly, performed well in most of the tested scenarios (Figures I.16c and I.17c). This result is not that surprising. If there is an optimal number of robots to find and exploit resources in a given environment, then swarms with long-range communication, where information spreads faster, reach that optimal number sooner and thus also become disadvantaged sooner as N_R increases.

I.9 Discussion

I.9.1 Summary of results

We have demonstrated that the nature of communication between individual robots affects a swarm's plasticity and thereby its ability to forage effectively in various types of environment. In particular, the following principles apply:

- Recruitment is most useful when deposits are large and hard to find. Conversely, recruitment can harm foraging performance when resource is distributed over numerous small deposits (see Figure I.4).
- Longer communication range increases foraging performance when deposits are large and hard to find (see Figure I.4), unless rapid exploitation of deposits can cause congestion, e.g., when a lot of the collected material accumulates in the base (see Figure I.15d) or when swarm size is large (see Figures I.16b,d and I.17b,d).
- If robots need to choose foraging sites based on deposit energy efficiency in *static* foraging environments, maximising information flow in the swarm is beneficial as it maximises exploitation of the best sources (see Figure I.5).
- When environments are *dynamic* and deposit qualities change over time, both continuous exploitation and exploration of the environment are important. Regulation of information flow, for example by short communication range of robots or by an implementation of individual-based decisions on when to ignore social information, is beneficial as it prevents overexploitation of a single foraging site (see Figures I.8c,d and I.9c,d). A slower information flow thus helps to avoid situations in which a swarm locks into foraging from a single deposit, or is unable

to discriminate deposits of varying qualities and forages from multiple sources simultaneously.

- Inhibition of social information spread can also be achieved when robots are to some extent physically prevented from meeting and recruiting each other, for example when the shape of the base prevents robots from communicating with those that approach from the opposite side (see online supplementary material, Figures S7 and S8), or when collected material accumulates in the base (see online supplementary material, Figures S11 and S12).

- On the other hand, information is collected and spreads faster when swarm size increases (see online supplementary material, Figures S13 and S14). This can be beneficial especially when robot communication is short range (see Figures I.16c and I.17a,c) or in scenarios where smaller swarms cannot alternate between deposits that change their qualities (see Figure I.11).

I.9.2 Related work

Our results are consistent with existing studies in the literature. For example, it has been shown that communication between robots is most beneficial when deposits are scarce (e.g., Liu et al., 2007a; Pitonakova et al., 2014) and that short-range communication is most suitable for large robot groups, as too much information can lower task specialisation (Sarker and Dahl, 2011). Previous work on the effects of communication range also demonstrated that global information exchange leads to over-commitment to a single resource (Tereshko and Loengarov, 2005). Similarly, our swarms of beggers, where information spreads quickly, often found themselves committed to a single source in scenarios with two deposits.

The importance of recruitment when flower patches have a high return has been observed in honey bee colonies (Donaldson-Matasci and Dornhaus, 2012). Bees, like our robots in Heap scenarios, can benefit from communicating about a location that affords many successful foraging trips. The fact that recruitment increases the probability of individuals foraging is also important in our Heap scenarios, where robots found it difficult to discover deposits. Similarly, in an agent-based model of a bee colony, recruitment was important in environments with few flower patches (Dornhaus et al., 2006).

A similar scenario has been explored by the HoFoReSim model (Schmickl et al., 2012), where simulated bees foraged from nectar sources, the quality of which changed once during a simulation run. The colony consisted of foragers that brought nectar into the nest and receivers that processed it. If a forager could not find a receiver to unload the nectar to, i.e., when the nest's nectar intake rate was too high for the receivers to handle, the forager did not waggle dance, but instead performed a tremble dance in order to activate additional dormant receivers. The authors argued that the ratio of

active receivers to foragers played an important role in the ability of the colony to switch between different sources of nectar when quality changes occurred. Foragers were able to switch to a better source of nectar more quickly when there were relatively few receivers. In this situation, successful foraging overloaded receivers, causing foragers to spend time activating more receivers rather than recruiting more foragers to their foraging sites. This reduction in forager recruitment allowed bees to be less committed to high quality foraging sites, making responses to a change in source quality faster. The role played by scarce receivers was therefore one of information regulation. In our simulation, regulation of information spread is achieved differently, e.g., by shorter communication range or constraints imposed on movement of the robots, but it has the same effect, allowing the swarms to respond to changes in the environment more appropriately. Under this reading, information flow considerations help explain a swarm's ability to exhibit plasticity both in our simulation and in HoFoReSim.

I.9.3 Swarm-level plasticity

In order to analyse foraging swarms, we quantified the value of information transfer and identified the following four work modes that a swarm could reach: switching, delayed switching, locked and indecisive. Swarms in the switching mode could respond to changes in deposit qualities well and usually performed better than swarms in other work modes (for example, see Figure I.8c,d, Figure I.9c,d and Figure I.9a for Heap4). Delayed switching, locked and indecisive modes were associated with worse levels of performance, although the extent of their disadvantage varied based on the environment and the swarm size.

Figure I.18 shows the work modes exhibited by swarms using the different behavioural strategies explored here, and the conditions under which they changed from one mode to another. The strategies are ordered from top to bottom by their ability to spread information, i.e., by the size of the positive information value region following the first deposit quality change. We tested the swarms under normal conditions, explored in Section 6, and under three kinds of perturbation (restricted dance floor shape, pellet accumulation and increase swarm size), explored in Section 8, giving us four foraging conditions in total. In addition, we explored four different scenario types: Heap2 vs. Heap4, and $T_Q = 1$ h vs. $T_Q = 2$ h. We could thus compare the behavioural strategies across a total of $4 \times 4 = 16$ different experimental setups (Figure I.11).

Short-range checkers were in the switching mode in all four scenario types under normal foraging conditions and exhibited the indecisive mode only when material accumulation increased in the Heap4 scenarios with $T_Q = 1$ h and $T_Q = 2$ h. We can say that the swarms were in the switching mode in 14 out of 16 (87.5%) of cases. Long-range checkers also found themselves in the switching mode in all scenarios under normal conditions, but there were three conditions under which they became either indecisive or delayed.

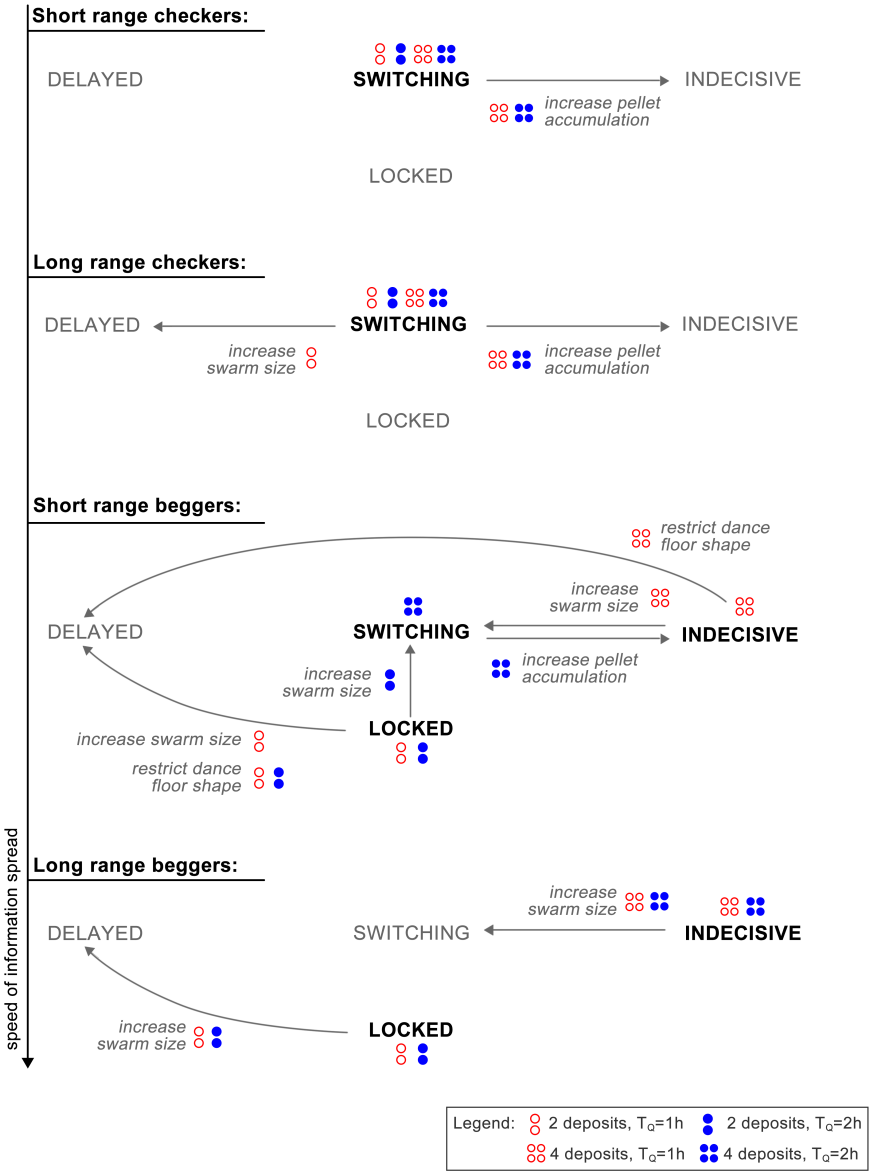


Figure I.18: Work modes and mode transitions exhibited by swarms with different behavioural strategies. The default modes that swarms operated in under normal conditions i.e., full dance floor, no pellet accumulation, 25 robots - see Sections I.5 and I.6) are shown in bold. Scenario markers related to a combination of the number of deposits and the length of deposit quality change interval, T_Q, are described in the figure legend. The markers are placed above default work modes and the transition arrows to indicate the scenarios in which they occurred.

These swarms were in the switching mode in 13 out of 16 (81.25%) cases. Moving along the information speed axis, short-range beggers were in the switching mode in only 5 out of 16 (31.25%) cases, and long-range beggers in only 2 out of 16 (12.5%) cases.

It is clear that rapid information spread often leads to the delayed, locked or indecisive modes, i.e., that it prevents plastic self-organisation. However, it is important to

point out that foraging performance in static environments was higher when information spread was fast. Furthermore, while the use of long-range recruitment in dynamic environments decreased the number of experimental setups where swarms were in the switching mode, it also led to a greater amount of resource being collected when the switching mode did occur and when congestion was not too high due to large swarm size or material accumulation. We could thus say that while using a robot behavioural strategy where information spreads quickly might be desirable in some application scenarios, designers of swarms have to be more certain about foraging conditions like swarm size, material accumulation, number of deposits, or the length of the deposit quality change interval. On the other hand, while they were never the best in terms of foraging performance, swarms of short-range checkers, where information spread slowly, exhibited plasticity more commonly, as they retained their ability to switch between deposits under most tested conditions. Such a strategy would thus be more suitable in unknown or uncontrollable environments.

These findings on the effects of information flow could be applied to a wider range of problems. In particular, the following swarm robotic tasks are the most relevant:

- A number of researchers are interested in collection from different deposit types at the same time (e.g., Balch, 1999; Campo and Dorigo, 2007). Results here show that inhibition of social information spread, which can be achieved by reducing communication range or restricting the opportunities for communication to take place (in our study, this is effected by restricting the shape of the recruitment area or altering the character of pellet accumulation), can lead to the desired outcome of foraging from different deposits in parallel.

- Communication about resource availability was used in robots that could adapt their resting and waking thresholds based on food density (Liu and Winfield, 2010). The speed of information spread, dependent on design decisions about communication type and range, could potentially play an important role in such a task and affect a swarm's flexibility.

- Task allocation has received substantial interest (e.g., Zhang et al., 2007; Sarker and Dahl, 2011; Jevtic et al., 2012; Zahadat et al., 2013). It is somewhat similar to deposit selection, as robots need to collectively explore the space of possible work sites and perform work on them. Principles of information transfer could be applied here, for example by substituting deposit energy efficiency for work site importance or utility during action selection and during transmission of social information. Similarly, information value, introduced in Section 7, could be adapted to relate to work site importance rather than deposit quality. We aim to explore this domain in our future work.

- As an alternative to design by hand, swarm behaviour design can be achieved by reinforcement learning (e.g., Pérez-Uribe, 2001) or artificial evolution (e.g., Doncieux et al., 2015; Ferrante et al., 2015). Knowing what types of communication strategies are beneficial under various conditions or at least which ones are more flexible could minimise the parameter search space for these optimisation algorithms and potentially decrease complexity of the resulting behaviours or save some simulation time.

Finally, it would be possible to utilise information value in an adaptive robot control algorithm in order to make a swarm more autonomous. For example, if robots could identify pathological effects of fast or slow information transfer, they could alter their individual behaviour or their communication range in order to achieve a more effective work mode.

I.9.4 Conclusion

Swarm foraging has many robotic applications and can also be used as a paradigm for other swarm behaviours such as task allocation, labour division or robot dispersion. Social insects like ants and bees give an example of how powerful swarm intelligence can be and how plastic adaptive behaviour can emerge from interactions between relatively limited system parts. However, before we can use robotic swarms in the real world, we need to understand how to design individual agents for reliable collaborative work in dynamic environments.

We argued that the way in which the character of robot-robot communication influences the flow of information through a swarm and the swarm's ability to obtain new information from the environment is an important factor that affects swarm plasticity and, as a consequence, foraging performance. Behavioural strategies that limit the spread of information promote plastic self-organisation and are thus more suitable for unknown or highly dynamic environments. We have also demonstrated a method for measuring the value of information that can help us identify such plastic behavioural strategies, not only in foraging swarms but also in a wider range of swarm behaviours where social information is utilised.

Appendix J

"Task allocation in foraging robot swarms: The role of information sharing"

This paper was published as Pitonakova, L., Crowder R. & Bullock, S. (2016). Task allocation in foraging robot swarms: The role of information sharing. In Gershenson, G. et al. (eds.), *Proceedings of the Fifteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE XV)*, MIT Press, 306–313

Abstract

Autonomous task allocation is a desirable feature of robot swarms that collect and deliver items in scenarios where congestion, caused by accumulated items or robots, can temporarily interfere with swarm behaviour. In such settings, self-regulation of workforce can prevent unnecessary energy consumption. We explore two types of self-regulation: *non-social*, where robots become idle upon experiencing congestion, and *social*, where robots broadcast information about congestion to their team mates in order to socially inhibit foraging. We show that while both types of self-regulation can lead to improved energy efficiency and increase the amount of resource collected, the speed with which information about congestion flows through a swarm affects the scalability of these algorithms.

J.1 Introduction

Congestion is an important factor that can negatively affect the performance of robot swarms (Hoff et al., 2010). Today's robotic systems, utilised in automated warehouses (D'Andrea, 2012), agriculture (Cartade et al., 2012), or in hospitals (Thiel et al., 2009),

must maintain effective work schedules for individual robots in order to minimise interference between robots and save energy. Decentralised task allocation, which affords redundancy and scalability, has been proposed as a possible solution (Wawerla and Vaughan, 2010; D'Andrea, 2012) that is likely to become more important in the near future as autonomous robot swarms will increasingly be deployed in unstructured and dynamic environments. In this paper, we explore the problem faced by a robot swarm that collects items from the environment and can cope with congestion by regulating its workforce in a decentralised manner in order to save energy. Furthermore, we explore how the means by which information about congestion is obtained by the robots affects the scalability of the swarm's performance under various foraging conditions.

During foraging, congestion can either result from the size of the robot population or from the structure of the environment. For example, robots might be required to wait until an occupied resource drop-off location becomes accessible (Wawerla and Vaughan, 2010) or they might need to queue in order to leave a crowded drop off location to perform more work (Krieger and Billeter, 2000). An autonomous robot swarm should be able to sense when congestion has become a problem and adjust its workforce accordingly.

We simulate robot swarms that collect items from the environment and drop them off in a central base, from where the items are consumed at a given rate. We explore two types of workforce self-regulation: *non-social*, where robots become idle upon directly experiencing severe congestion, and *social*, where a robot will inhibit the foraging of its team mates by signalling them to become idle when the congestion that it experiences is severe. We show that both types of self-regulation can lead to significant energy savings and thus to a greater number of items collected when the energy available to the robots is limited. More importantly, we demonstrate that the speed with which information about congestion flows through a swarm, either when robots detect congestion themselves or when they exchange information with their team mates, affects the swarm's ability to respond to it appropriately. While social self-regulation results in rapid information flow and can lead to significant performance benefits in certain scenarios, it can also lead to significantly worse performance in others. On the other hand, non-social self-regulation, where information flow is slower, leads to improved energy efficiency across a greater number of foraging conditions, making it more suitable in unknown environments, although it is outperformed by social self-regulation in some cases.

The following sections provide an overview of related work and a description of our simulation and analysis methods. We then compare, across a number of experimental scenarios, the performance of our two types of self-regulated swarms with that of control swarms that do not use self-regulation. We evaluate both the amount of energy needed to collect items and the number of items collected when robot energy is limited. We conclude with a discussion of how our results relate to our previous work on information flow in swarms (Pitonakova et al., 2016a) and provide examples of real-world applications where the two types of self-regulated swarms could be used.

J.2 Related Work

Route planning systems that optimise robot traffic are often used in controlled warehouse environments with small robot teams (Vivaldini et al., 2010; Mather and Hsieh, 2012). However, such approaches require a centralised controller to guide robot behaviour and are thus unsuitable for large swarms, where computation of optimal solutions becomes infeasible (Dahl et al., 2009). Furthermore, a model of the environment and of the tasks within it, which centralised planning systems rely on, can be difficult to obtain in dynamic or unstructured environments. On the other hand, decentralised decision making, where robots change their behaviour based on limited local information obtained through their sensors, is more suitable for complex tasks of this type (Hoff et al., 2010).

Decentralised robot decision making has been well studied in a number of logistic and foraging applications. For example, unmanned vehicles that need to transport items from one location to another can adjust their work time and decide to recruit others based on the number of items in pick-up locations (Wawerla and Vaughan, 2010). In behaviour-based robotics, a combination of environmental cues, such as the presence of items or other robots nearby, can trigger or inhibit foraging behaviour, leading to self-organised division of labour between robots that are idle and those that collect resources (Jones and Mataric, 2003). Alternatively, 'bucket brigading' robots can form chains of work areas and progressively transport items between two locations (Shell and Mataric, 2006; Pini et al., 2013) and even adapt the size of the work areas based on collisions with other robots in order to improve their performance (Lein and Vaughan, 2009).

The *Response Threshold Model* (RTM) is a self-regulatory mechanism inspired by social insects (Bonabeau et al., 1997) that has been applied in a number of simulated and real-world robot experiments. According to the model, robots alternate between *foraging* and *resting* based on some internal, environmental or social cues in order to optimise their energy consumption. For example, robots can count the number of items stored in the base and only leave to forage when the number is below a specified threshold (Yang et al., 2009). Robots can also evaluate how many items they encountered during foraging and decide to rest if the environment is not rich enough (Labella et al., 2006). By counting the number of collisions with other robots (Liu et al., 2007b) or by detecting drops in their own expected performance (Dahl et al., 2009), robots can decide to rest if they estimate that congestion is beyond an acceptable level. Finally, in dynamic environments, where the number of items in the environment changes over time, robots can decrease the energy cost of collecting items by only foraging when enough items are available, estimating the state of the environment in either a centralised (Liu et al., 2007b) or decentralised (Dai, 2009) manner.

Our work builds on the Response Threshold Model literature, and in particular on the work of (Liu et al., 2007b) and (Dai, 2009), where robots estimated the level of congestion in order to prevent unnecessary energy consumption. However, we apply the RTM in a

novel scenario, where successful foragers *recruit* other robots to the worksites that they are exploiting (see also Pitonakova et al., 2014,0).

Furthermore, we provide novel insights into the role played by information flow in decentralised congestion estimation. In our non-social model, congestion is estimated by each robot individually, while in the social model, robots communicate their estimates to nearby robots in order to socially inhibit foraging.

Our approach to congestion estimation is inspired by the self-regulatory behaviour of honey bees foraging for nectar (Anderson and Ratnieks, 1999; Gregson et al., 2003). When nectar is abundant, foragers may bring more nectar into the hive than the nectar-receiving bees can cope with. In order to prevent unnecessary foraging, individual foraging bees evaluate how long it takes for their nectar to be unloaded. If unloading is taking too long, a forager will *tremble dance* around the nest, inhibiting other bees from recruiting and thus reducing the number of foragers. Our social RTM uses a similar principle.

J.3 Methods

J.3.1 Environment

All our experiments are performed in the ARGoS simulation environment which implements realistic 3D physics and robot models (Pinciroli et al., 2012). The simulation has continuous space and updates 10 times per simulated second. A circular base with a diameter of 3 metres is situated in the centre of the experimental arena. The base is divided into three sections (Figure J.1): an interior circular *resting bay* with an annular *observation bay* around it and an annular *unloading bay* around that. A light source, placed above the centre of the base, is used by robots as a reference for navigation towards and away from the base centre (as in, e.g., Krieger and Billeter, 2000; Pini et al., 2013).

Cylindrical resource deposits with radius r_D are placed outside of the base, each containing an unlimited volume of resource. In order to enable robots close to a deposit to move towards it, a colour gradient with radius r_C is present on the floor around each deposit.

We explore two types of scenarios:

- HeapN: $N \leq 4$ deposits distributed evenly around the base at a distance $D = \{5, 7, 9\}$m from the base edge. These deposits represent large heaps of resource (e.g., mineral deposits), with $r_D = 0.5$m and $r_C = 3$m.

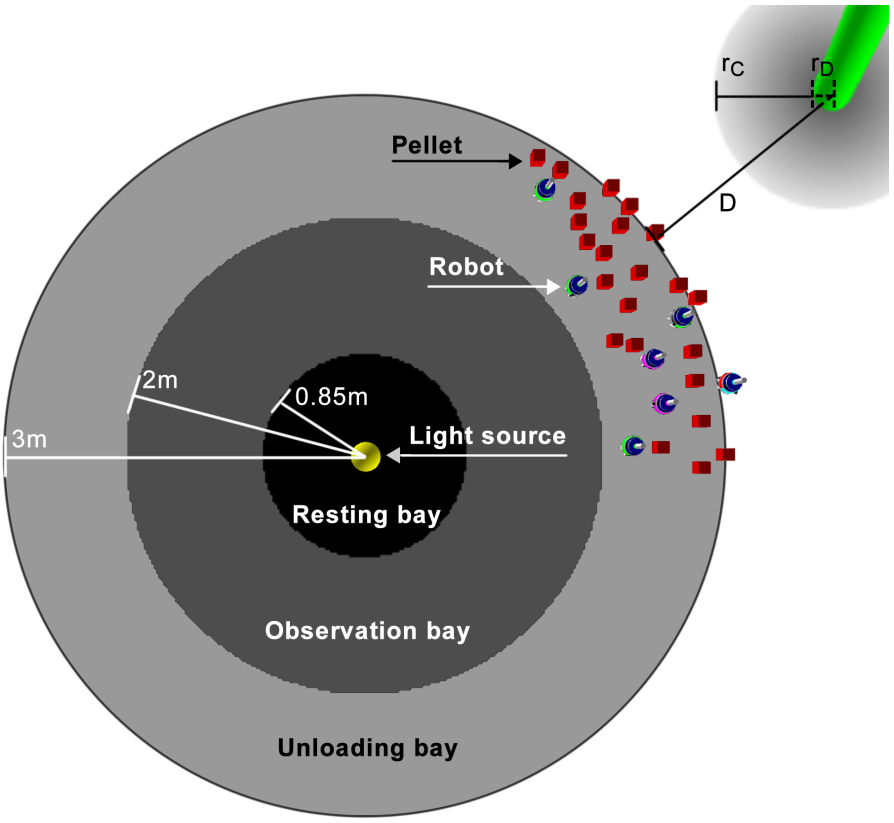


Figure J.1: ARGoS simulation screenshot of the base and a deposit Dm away from the base edge. The base consists of a resting bay, an observation bay and an unloading bay. A light source is placed above the centre of the base to guide robot navigation. Pellets collected by the robots temporarily accumulate in the unloading bay, causing congestion.

- ScatterN: $N \geq 10$ deposits randomly distributed between $D-5$m and $D+5$m from the base edge. These deposits are small (e.g., litter on a street), with $r_D = 0.1$m and $r_C = 1$m.

J.3.2 Robots

The simulated MarXbots (Bonani et al., 2010) are circular, differentially steered robots with a diameter of 0.17m that can reach a maximum speed of 5cm/s in our simulation. The robots use infra-red sensors for obstacle avoidance and communication, colour sensors for navigation towards nearby deposits, and a light sensor for phototaxis towards the base (see Pitonakova et al., 2016a, for more details). The robots are modelled as finite-state machines and can implement three types of homogeneous swarm: *control swarm*, *non-social* self-regulators, and *social* self-regulators (Figure J.2).

Control swarm robots exhibit basic foraging behaviour with no self-regulation. A robot starts with a random orientation and a random position in the observation bay as an

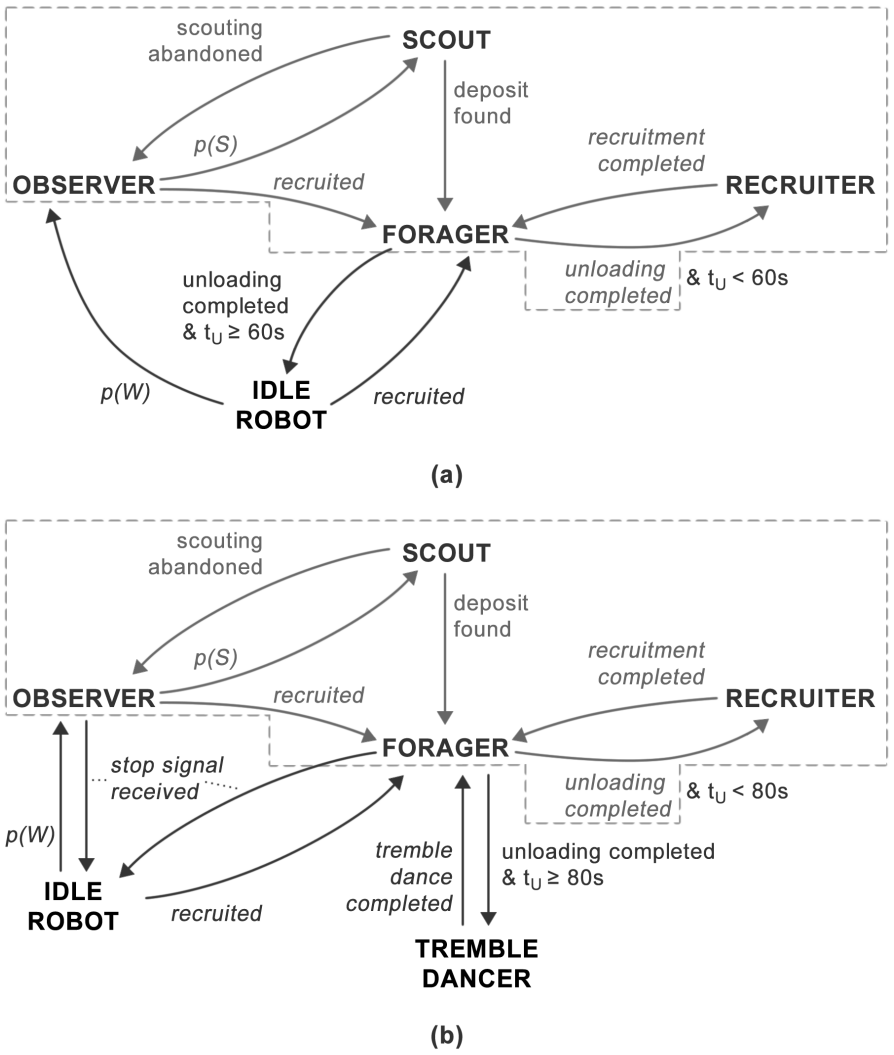


Figure J.2: Finite state machine representation of the robot controller in swarms with (a) non-social self-regulation, and (b) social self-regulation. The behaviour of the control swarm controller is displayed in dashed boxes.

observer, ready to receive and follow recruitment signals. An observer moves randomly across the observation bay and avoids traveling into the unloading and resting bays. At each time step an observer can become a *scout* with scouting probability $p(S) = 10^{-3}$. A scout leaves the base and uses Lévy movement (Reynolds and Rhodes, 2009) to search for a resource deposit within 20m of the base. The robot updates its estimated location relative to the base using path integration based on odometry at each time step (e.g., Lemmens et al., 2008; Gutiérrez et al., 2010). When a deposit is found, the robot loads one unit of volume of resource and returns back to the base utilising phototaxis, while keeping track of its position relative to the deposit using odometry. Odometry noise is not modelled. Any scout that cannot find a deposit within 600s returns to the base and becomes an observer.

A laden robot returning to the base drops off its load in the unloading bay in the form of

four pellets of size 0.1m^3. The robots cannot push existing pellets around and thus have to avoid them in order to traverse the unloading bay. A new pellet can only be deposited when there is enough free space in front of the robot. Deposited pellets disappear from the simulation (representing their utilisation by a hypothetical unmodelled system of robots or human users) after a period of *unloading bay handling time*, t_H. When $t_H = 1\text{s}$, pellets disappear very quickly and do not cause congestion. By increasing the value of t_H, we can experiment with the level of congestion in the simulation, as more accumulated pellets make entering and leaving the base more difficult.

After depositing the pellets, the robot moves further into the base and performs recruitment for 120s, randomly moving across the base while avoiding re-entering the unloading bay. A recruiter advertises the fact that it has information about a deposit to all observers located within recruitment range of 0.6m. Deposit location is communicated to each observer in a one-to-one fashion by taking into account the local axes of the robots and their alignment relative to each other (Gutiérrez et al., 2010). The recruiter resumes foraging from the same deposit after it completes recruitment.

In *self-regulated swarms*, robots additionally measure their *pellet unloading time*, t_U, i.e., the time between entering the base and leaving the unloading bay. Robots in swarms with *non-social self-regulation* (Figure J.2a) proceed to the resting bay and become *idle* after depositing pellets if congestion is severe (i.e., $t_U \geq 60\text{s}$). An idle robot consumes a negligible amount of energy (as in, e.g., Wawerla and Vaughan, 2010) and can be woken up and immediately recruited by a recruiter, i.e., by a robot that does not experience severe congestion. In order to avoid deadlocks, an idle robot can also wake up spontaneously with a waking probability $p(W) = 10^{-4}$.

Robots in swarms with *social self-regulation* (Figure J.2b) do not become idle after experiencing severe congestion (i.e., when $t_U \geq 80\text{s}$), but become *tremble dancers* instead. A tremble dancer travels randomly across the observation bay and broadcasts *stop signals* to all robots within a 0.9m range for 120s, after which it leaves the base to resume foraging without recruiting. Stop signals inhibit foraging as any observer or forager that receives a stop signal moves to the resting bay and becomes idle. Stop signals also inhibit recruitment, as they cause any recruiter in range to cease recruiting and immediately leave the base to forage.

J.3.3 Analysis

We performed 50 simulation runs that lasted 4 simulated hours in Heap1, Heap4, Scatter10 and Scatter25 scenarios and compared the performance of all three swarm types using $N_R = 25$ and $N_R = 50$ robots. Since we are interested in efficient energy usage, we define a performance metric, *energy efficiency*, C, which represents the amount of energy a swarm spends in order to collect a unit of resource:

$$C = \frac{R}{E} \tag{J.1}$$

where R is the total amount of resource collected by the swarm and E is the total amount of energy expended by the swarm. It is assumed that an idle robot expends 0 units of energy per second and a robot in any other state expends 1 unit of energy per second. Since the control swarm robots are never idle, control swarms spend a total of $N_R \times (4 \times 60 \times 60) = N_R \times 14,400$ units of energy in each 4-hour experiment. We compare C values achieved by the two types of self-regulated swarms with that achieved by the control swarms in order to find out how advantageous self-regulation was in different scenarios.

We also analyse how much resource the swarms collected when energy availability was limited. During this analysis, it is assumed that all robots stop working when the swarm spends $N_R \times E'$ units of energy, where E' is the energy limit per robot. Energy limits may play a role for example in planet exploration, where robots might use a common solar-powered energy repository of a limited capacity.

J.4 Simulation Results

In the following sections, we compare the control swarms to each of the two kinds of the self-regulated swarms in terms of their energy efficiency, C, and the amount of resource they collected, R. We show that the self-regulated swarms can achieve better C in scenarios where pellets cause significant congestion. Furthermore, self-regulation leads to a higher amount of resource collected when the total energy available to the swarms is limited in such scenarios. We also discuss cases when self-regulation leads to performance deterioration, especially when social self-regulation is used.

J.4.1 Energy efficiency

In this section we report the performance (in terms of energy efficiency) of different swarm types in each of 48 scenarios: 2 swarm sizes (25 and 50) × 2 unloading bay handling times (5s and 20s) × 3 deposit distances (5m, 7m , and 9m) × 4 deposit distribution types (Heap1, Heap4, Scatter10 and Scatter25). In each case, we report the average performance of 50 self-regulated swarms relative to the average performance of 50 control swarms in the same scenario.

First we will summarise the performance of the control swarms, depicted in Figure J.3. Their resource collection performance was more attenuated by congestion when the number of robots was large ($N_R = 50$) and when unloaded pellets did not disappear quickly from the unloading bay ($t_H = 20$s). Congestion was especially problematic in scenarios

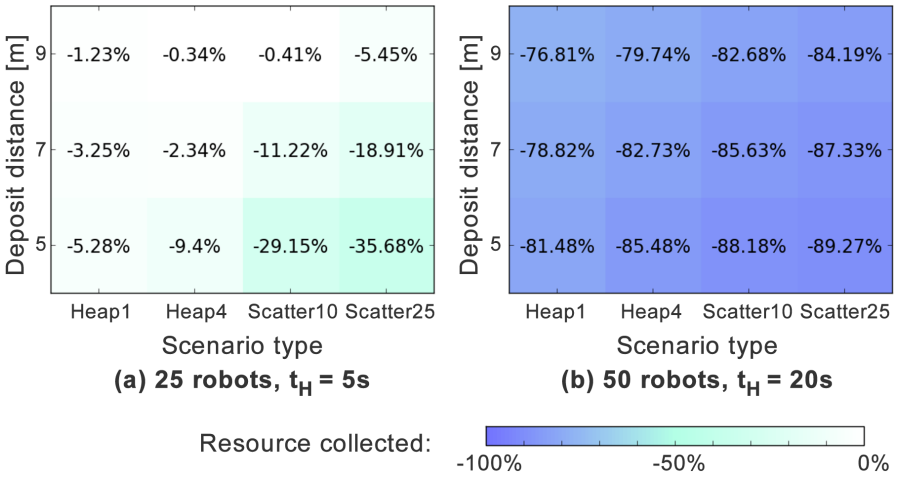


Figure J.3: Resource collection performance of control swarms relative to experiments with no congestion (i.e., when $t_H = 1$s).

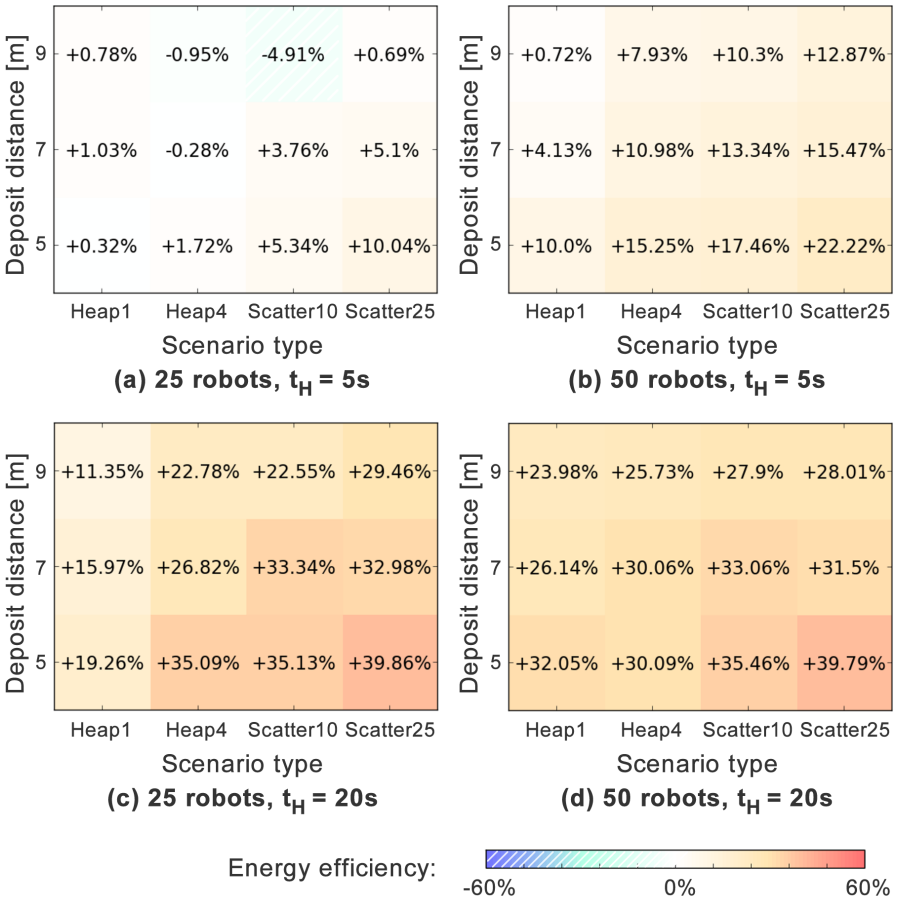


Figure J.4: Performance of non-social self-regulated swarms relative to control swarms.

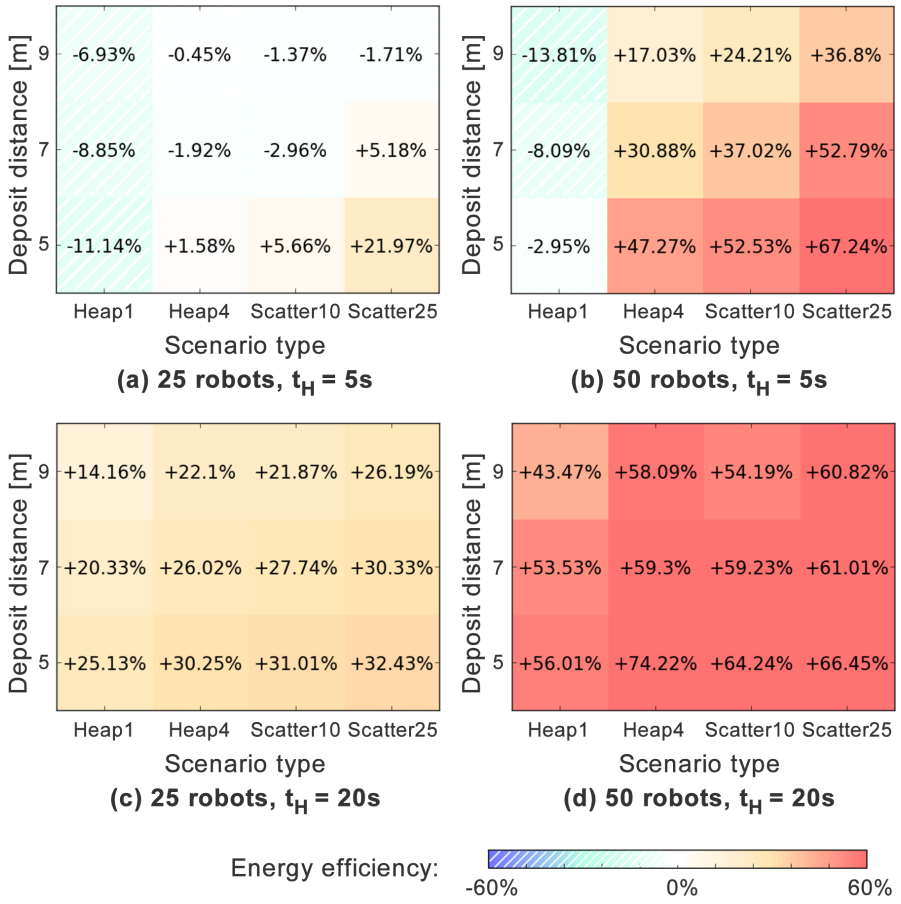


Figure J.5: Performance of social self-regulated swarms relative to control swarms.

with a large number of deposits and when deposits were closer to the base. More severe performance deterioration was measured in the Scatter scenarios, where multiple foraging locations were exploited at the same time, causing fast pellet accumulation around the whole unloading bay.

Evaluating the performance of *non-social* self-regulated swarms relative to that of control swarms (Figure J.4) indicates that non-social self-regulators tended to enjoy more of an advantage when control swarms were more affected by congestion. Consequently, where congestion was very mild (e.g., a small number of robots foraging for a few heaped deposits distributed far from a base that handles unloaded deposits quickly), the performance of non-social swarms and control swarms was very similar ($\approx \pm 1\%$ difference), and control swarms even enjoyed a 5% advantage in the mildest Scatter10 scenario. However, where congestion tended to be severe (e.g., a large number of robots foraging for many scattered deposits distributed near to a base that handles unloaded deposits slowly), the performance of non-social swarms was considerably greater than that of control swarms (up to $\approx +40\%$ in the most extreme Scatter25 environments).

The performance of *social* self-regulatory swarms relative to the control swarms follows a similar but more complicated pattern (Figure J.5). Again, where congestion tended to be severe, the performance of social swarms was better than that of control swarms (up to $\approx +67\%$ in the most extreme Scatter25 environments). Moreover, in these scenarios, social swarms did even better than non-social swarms, achieving an advantage over the control swarms that was often between 20% and 40% larger than that achieved by non-social swarms. Conversely, in scenarios where congestion was very mild, the performance of social swarms was *worse* than that of control swarms and non-social swarms by as much as -14%.

In general, there were two factors that affected the advantage of self-regulation: the amount of congestion in the base and the distribution of deposits in the environment. For instance, self-regulation was most advantageous in Scatter scenarios when deposits were close to the base (i.e., when the control swarms experienced high congestion due to short trips between the base and the deposits), and, more importantly, when foraging effort could be refocussed in a new direction once a particular part of the unloading bay became congested. On the other hand, self-regulation was not as effective in the Heap1 scenarios, where all resources were concentrated in a single location. Robots in the self-regulated swarms could still become idle when pellets accumulated, but recruitment could only take place again when the foraging robots measured a low unloading time, i.e., when enough of the pellets that had been unloaded in the part of the unloading bay between the deposit heap and the resting bay had disappeared. This was a particular problem for the swarms with social self-regulation, where the information about congestion spread quickly through the swarm, causing a majority of the robots to become idle. Unlike in non-social swarms, the number of foraging robots was often very low and it took the social swarms a long time to recover from inactivity.

J.4.2 Resource collection

In this section we report the average performance (now in terms of total amount of resource collected) of the self-regulated swarms relative to the average performance of the control swarms. As in the previous section, we consider cases with mild congestion ($N_R = 25, t_H = 5s$) and severe congestion ($N_R = 50, t_H = 20s$). We first consider experiments where the total energy available to the swarms was unlimited. We then report on experiments with energy limitation, where robots ceased foraging as soon as their swarm had consumed $N_R \times E'$ units of energy, where E' was the energy limit per robot. Figure J.6 depicts our results.

Although both types of self-regulated swarms were often more energy efficient than control swarms, when energy was unlimited they did not tend to collect more resource than the control swarms. Non-social self-regulators tended to collect a similar quantity

to control swarms, whereas social self-regulators collected less resource than control swarms when congestion was mild (Figure J.6a,c)

When swarm energy was limited, non-social self-regulators tended to either collect significantly more resource than control swarms (when congestion was severe), or roughly the same amount as control swarms (when congestions was mild). For instance, when E' was set to 8000 and when a large number of robots foraged from a base that handled unloaded deposits slowly in the Scatter25 scenario, non-social swarms foraged up to 30% more resource relative to control swarms (Figure J.6b). In experiments where congestion was mild, the advantage of non-social swarms was less pronounced. For instance, when a small number of robots foraged from a base that handled unloaded deposits quickly, non-social self-regulated swarms only collected up to 10% more resource than control swarms (Figure J.6a).

Social self-regulation again led to more extreme variation in performance when the swarm

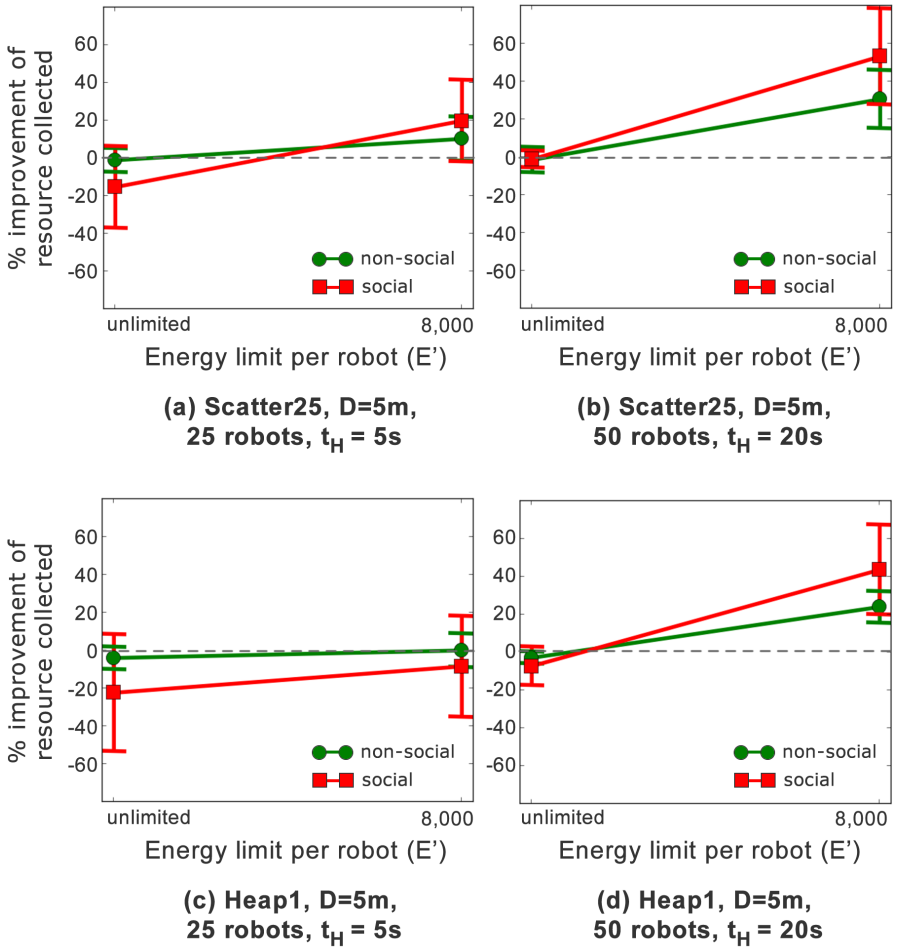


Figure J.6: Resource collection performance of self-regulated swarms relative to control swarms under unlimited and limited energy conditions for scenarios with (a, c) mild congestion, and (b, d) severe congestion.

energy was limited. When congestion was severe, social self-regulators tended to collect significantly more resource than either control swarms or non-social self-regulators (Figure J.6b and J.6d). Whereas when congestion was mild, they collected roughly the same amount as control swarms and social self-regulators (Figure J.6a and J.6c). For instance, in Scatter25, social-self-regulators collected approximately 55% more resource than the control swarms when $E' = 8000$ (Figure J.6b). On the other hand, in Heap1, where the robots could not spread their foraging effort to other directions once a particular part of the unloading bay became congested, social self-regulators collected on average 10% *less* resource than the control swarms when congestion was mild (Figure J.6c). In both cases, variation in performance within a scenario was higher for social swarms.

When the value of E' was higher or lower than 8000, the relative performance of both self-regulated swarms decreased linearly but was never lower than when the swarm energy was unlimited.

J.5 Discussion and Conclusions

We have shown that swarms can regulate their foraging activity effectively on the basis of locally perceived levels of congestion. The solution presented in this paper extends previous studies of the Response Threshold Model (RTM) (e.g., Liu et al., 2007b; Dahl et al., 2009; Yang et al., 2009), applying it for the first time to foraging swarms that use recruitment and investigating the effect of information sharing during decentralised congestion estimation.

We compared three types of swarms: control swarms with no self-regulation, swarms with non-social self-regulation (where robots become idle when they directly sense severe congestion), and swarms with social self-regulation (where robots instruct their team mates to become idle when they detect severe congestion). The swarms were assessed across a number of experimental scenarios, where we varied the number of deposits (N_D), deposit distance from the base (D), the number of robots (N_R), and the time it took for accumulated material to be consumed at the base (t_H). We evaluated the performance of the swarms in terms of energy efficiency, C, and showed that C can be improved through self-regulation especially in environments where the collected material accumulates in the base quickly (because D is small, or N_R or t_H are large) or where the swarms can exploit multiple foraging directions simultaneously (i.e., because N_D is large). Additionally, we demonstrated that self-regulated swarms collect more resources than control swarms when the energy supply available to the robots is limited.

There were notable differences in how swarms with non-social and social self-regulation performed in the various experimental scenarios. By comparison with control swarm behaviour, non-social self-regulation led to mediocre performance improvements or equivalent levels of performance in some scenarios. On the other hand, social self-regulation

achieved large improvements over control swarms in scenarios where pellets accumulated quickly, but were also outperformed by control swarms in scenarios where congestion was not as severe or where all resources were concentrated in a single location. In our work on information flow in foraging swarms that use recruitment (Pitonakova et al., 2016a), we argued that *fast* information flow can lead to pathological states of a whole swarm that prevent the swarm from responding to changes in the environment. Furthermore, we demonstrated that while swarms with fast information flow tend to perform extremely well in a limited number of environments but perform poorly in others, swarms with *slow* information flow tend to perform well across a broad spectrum of scenarios. In this paper, we extend this argument to scenarios involving congestion. Information flow was slower in swarms of non-social self-regulators which relied on their own local perception alone, and it was faster in swarms of social self-regulators which communicated information about congestion to one another. As was the case in (Pitonakova et al., 2016a), slow information flow led to behaviour suitable for a larger number of experimental scenarios, while fast information flow caused more extreme variation in performance meaning it was only appropriate in a restricted set of scenarios.

Consequently, robots inspired by our social self-regulated swarms could be applied effectively in appropriate well-defined foraging or logistic tasks, for example to deliver items between various locations in warehouses and hospitals, or to collect crops. In these scenarios, the relevant task parameters (swarm size, processing time of collected items, etc.) are known upfront. However, if we were to employ robot swarms in an unknown or more variable environment, e.g., work sites on different planets or underwater, we would need to take into account the fact that while fast information flow can lead to beneficially fast response times, it can also cause significantly suboptimal performance under certain conditions. In such applications, self-regulation that is more subtle and occurs in a more localised fashion would be more suitable, not because of the ability of the swarms to perform work faster or more efficiently, but because such collective behaviour is more scalable. It might also be advantageous to create an adaptive algorithm, where robots alter their own self-regulatory behaviours (for example their willingness to exchange information with others, their waking up probability, etc.), in order to achieve a level of information flow within the swarm that varies dynamically in a way that is appropriate to the swarm's current environment.

References

Alers, S., Bloembergen, D., Hennes, D., Jong, S. D., Kaisers, M., et al. (2011). Bee-inspired foraging in an embodied swarm. In Tumer, K., Yolum, P., Sonenberg, L., et al., editors, *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 1311–1312, New York. ACM.

Amazon Prime Air (2016). Available on https://www.amazon.com/b?node=8037720011 [Accessed on 04 Sep 2016].

Anderson, C. and Ratnieks, F. L. W. (1999). Worker allocation in insect societies: Coordination of nectar foragers and nectar receivers in honey bee (*Apis mellifera*) colonies. *Behavioral Ecology and Sociobiology*, 46(2):73–81.

Arab, A., Carollo Blanco, Y., and Costa-Leonardo, A. M. (2012). Dynamics of foraging and recruitment behavior in the asian subterranean termite coptotermes gestroi (Rhinotermitidae). *Psyche: A Journal of Entomology*, 2012:Article ID 806782.

Aridor, Y. and Lange, D. (1998). Agent design patterns: Elements of agent application design. In Sycara, K. P. and Wooldridge, M., editors, *Proceedings of the Second International Conference on Autonomous Agents (AGENTS '98)*, pages 108–115, New York. ACM.

Arkin, R. C. (1992). Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364.

Astrobotic (2015). Available on https://www.astrobotic.com [Accessed on 04 Sep 2016].

Bailis, P., Nagpal, R., and Werfel, J. (2010). Positional communication and private information in honeybee foraging models. In Dorigo, M., Birratari, M., Di Caro, G. A., et al., editors, *ANTS 2010*, pages 263–274. Springer.

Balch, T. (1999). The impact of diversity on performance in multi-robot foraging. In Etzioni, O., Müller, J. P., and Bradshaw, J. M., editors, *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 92–99, New York. ACM.

Balch, T. and Arkin, R. C. (1994). Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52.

Barbani, L. E. (2003). *Foraging Activity and Food Preferences of the Odorous House Ant (Tapinoma sessile Say)*. PhD thesis, Virginia Polytechnic Institute. Available on http://scholar.lib.vt.edu/theses/available/etd-06282003-114935/unrestricted/Laura_Barbani.pdf [Accessed on 04 Sep 2016].

Becker, A., Habibi, G., Werfel, J., Rubenstein, M., and McLurkin, J. (2013). Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, pages 520–527.

Beekman, M. (2001). Phase transition between disordered and ordered foraging in Pharaoh's ants. *PNAS*, 98(17):9703–9706.

Bel'kovitch, V. M. and Sh'ekotov, M. N. (1993). *The Belukha whale: Natural behaviour and bioacoustics*. Woods Hole Oceanographic Institution, Woods Hole, MA.

Benaroya, H., Bernold, L., and Chua, K. (2002). Engineering, design and construction of lunar bases. *Journal of Aerospace Engineering*, 15(2):33–45.

Benoit-Bird, K. J. and Au, W. W. L. (2009a). Cooperative prey herding by the pelagic dolphin, *Stenella longirostris*. *The Journal of the Acoustical Society of America*, 125(1):125–37.

Benoit-Bird, K. J. and Au, W. W. L. (2009b). Phonation behavior of cooperatively foraging spinner dolphins. *The Journal of the Acoustical Society of America*, 125(1):539–46.

Berthouze, L. and Lorenzi, A. (2008). Bifurcation angles in ant foraging networks: A trade-off between exploration and exploitation? *From Animals to Animats 10*, pages 113–122.

Bhuiyan, J. (2016). The complete timeline to self-driving cars. Recode. Available on http://www.recode.net/2016/5/16/11635628/self-driving-autonomous-cars-timeline [Accessed on 04 Sep 2016].

Biesmeijer, J. C. and De Vries, H. (2001). Exploration and exploitation of food sources by social insect colonies: A revision of the scout-recruit concept. *Behavioral Ecology and Sociobiology*, 49(2):89–99.

Bikhchandani, S., Hirshleifer, D., and Welch, I. (1992). A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of Political Economy*, 100(5):992 – 1026.

Bliege Bird, R., Smith, E. A., and Bird, D. W. (2001). The hunting handicap: Costly signaling in human foraging strategies. *Behavioral Ecology and Sociobiology*, 50(1):9–19.

Boeglin, J. (2015). The cost of self-driving cars: Reconciling freedom and privacy with tort liability in autonomous vehicle regulation. *Yale Journal of Law & Technology*, 17(1):171–203.

Bonabeau, E., Sobkowski, A., Theraulaz, G., and Deneubourg, J.-L. (1997). Adaptive task allocation inspired by a model of division of labor in social insects. In Lundh, D., Olsson, B., and Narayanan, A., editors, *Biocomputing and Emergent Computation: Proceedings of BCEC97*, pages 36–45, London. World Scientific Publishing.

Bonani, M., Longchamp, V., Magnenat S., Philippe, R., Burnier, D., et al. (2010). The MarXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, pages 4187 – 4193, Piscataway, NJ. IEEE Press.

Borenstein, J. (1998). Experimental results from internal odometry error correction with the OmniMate mobile robot. *IEEE Transactions on Robotics and Automation*, 14(6):963–969.

Bosch, J. (1994). Relations as Object Model Components. *Journal of Programming Languages*, 4:39–61.

Brambilla, M., Brutschy, A., Dorigo, M., and Birattari, M. (2014). Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Transactions on Autonomous and Adaptive Systems*, 9(4):Article no. 17.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

Brazier, F. M. T., Jonker, C. M., and Treur, J. (2002). Principles of component-based design of intelligent agents. *Data and Knowledge Engineering*, 41(1):1–27.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):114–23.

Brooks, R. A. (1989). Fast, Cheap, and out of Control: A Robot Invasion of the Solar System. *Journal of the British Interplanetary Society*, 42:478–485.

Brooks, R. A., Maes, P., Mataric, M. J., and More, G. (1990). Lunar base construction robots. In *Proceedings of the 1990 IEEE International Workshop on Intelligent Robots and Systems (IROS '90)*, pages 389–392.

Busse, C. (1978). Do chimpanzees hunt cooperatively? *The American Naturalist*, 112(986):767–770.

Campbell, A. and Wu, A. S. (2011). Multi-agent role allocation: Issues, approaches, and multiple perspectives. *Autonomous Agents and Multi-Agent Systems*, 22(2):317–355.

Campo, A. and Dorigo, M. (2007). Efficient multi-foraging in swarm robotics. In Almeida e Costa, F., editor, *Proceedings of the 9th European Conference on Advances in Artificial Life (ECAL 2007)*, pages 696–705, Berlin. Springer.

Campo, A., Gutiérrez, Á., Nouyan, S., Pinciroli, C., Longchamp, V., et al. (2010). Artificial pheromone for path selection by a foraging swarm of robots. *Biological Cybernetics*, 103(5):339–352.

Cartade, P., Lenain, R., Thuilot, B., Benet, B., and Berducat, M. (2012). Motion control of a heterogeneous fleet of mobile robots: Formation control for achieving agriculture task. In *Proceedings of the International Conference on Agricultural Engineering (CIGR-AgEng '12)*.

Castello, E., Yamamoto, T., Libera, F. D., Liu, W., Winfield, F. F. T., et al. (2016). Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach. *Swarm Intelligence*, 10(1):1–31.

Charnov, E. L. (1976). Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9(2):129–136.

Chong, K. S. and Kleeman, L. (1997). Accurate odometry and error modelling for a mobile robot. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 2783–2788, Piscataway, NJ. IEEE Press.

Clark, C. W. and Mangel, M. (1984). Foraging and flocking strategies: Information in an uncertain environment. *The American Naturalist*, 123(5):626 – 641.

Clark, C. W. and Mangel, M. (1986). The evolutionary advantages of group foraging. *Theoretical Population Biology*, 30(1):45–75.

Couceiro, M. S., Rocha, R. P., Figueiredo, C. M., Luz, J. M. A., and Ferreira, N. M. F. (2012). Multi-robot foraging based on Darwin's survival of the fittest. *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*.

Couzin, I. D. (2009). Collective cognition in animal groups. *Trends in Cognitive Sciences*, 13(1):36–43.

Dahl, R. S., Mataric, M. J., and Sukhatme, G. S. (2009). Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57:674–687.

Dai, H. (2009). Adaptive control in swarm robotic systems. *The Hilltop Review*, 3(1):54–67.

D'Andrea, R. (2012). Guest editorial: A revolution in the warehouse: A retrospective on Kiva Systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639.

D'Andrea, R. (2014). Guest editorial: Can drones deliver? *IEEE Transactions on Automation Science and Engineering*, 11(3):647–648.

Darimont, C. T., Reimchen, T. E., and Paquet, P. C. (2003). Foraging behaviour by gray wolves on salmon streams in coastal British Columbia. *Canadian Journal of Zoology*, (81):349–353.

De Marco, R. and Farina, W. M. (2001). Changes in food source profitability affect the trophallactic and dance behavior of forager honeybees (*Apis mellifera L.*). *Behavioral Ecology and Sociobiology*, 50(5):441–449.

De Marco, R. and Farina, W. M. (2003). Trophallaxis in forager honeybees *Apis mellifera*: Resource uncertainty enhances begging contacts? *Journal of Comparative Physiology A*, 189:125–134.

De Vries, H. and Biesmeijer, J. C. (2002). Self-organization in collective honeybee foraging: emergence of symmetry breaking, cross inhibition and equal harvest-rate distribution. *Behavioral Ecology and Sociobiology*, (51):557–569.

De Wolf, T. and Holvoet, T. (2007). Design patterns for decentralised coordination in self-organising emergent systems. In Brueckner, S. A., Hassas, S., Jelasity, M., and Yamins, D., editors, *Proceedings of the 4th International Workshop on Engineering Self-Organising Systems (ESOA'06)*, volume 4335 of *Lecture Notes in Computer Science*, pages 28–49. Springer, Berlin.

Dekker, A. (2006). Centralisation vs self-synchronisation: An agent-based investigation. In *Proceedings of the 11th International Command Control Research and Technology Symposium (ICCRTS)*, Washington, DC. CCRP Press.

Delton, A. W. and Robertson, T. E. (2012). The social cognition of social foraging: Partner selection by underlying valuation. *Evolution and Human Behavior*, 33(6):715–725.

Deugo, D., Weiss, M., and Kendall, E. (2001). Reusable patterns for agent coordination. In Omicini, A., Zambonelli, F., Klusch, M., and Tolksdorf, R., editors, *Coordination of Internet agents*, pages 347–368. Springer, Berlin.

Do, T. T., Kolp, M., and Pirotte, A. (2003). Social patterns for designing multi-agent systems. In Webb, G. and Dai, H., editors, *Proceedings of the 15th International Conference on Software Engineering & Knowledge Engineering (SEKE 2003)*, pages 103–110, Skokie, Ill. Knowledge Systems Instittute.

Donaldson-Matasci, M. C. and Dornhaus, A. (2012). How habitat affects the benefits of communication in collectively foraging honey bees. *Behavioral Ecology and Sociobiology*, 66(4):583–592.

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. (2015). Evolutionary robotics: What, why, and where to. *Frontiers in Robotics and AI*, 2(4):DOI: 10.3389/frobt.2015.00004.

Dornhaus, A., Klügl, F., Oechslein, C., Puppe, F., and Chittka, L. (2006). Benefits of recruitment in honey bees: Effects of ecology and colony size in an individual-based model. *Behavioral Ecology*, 17(3):336–344.

Drogoul, A. and Ferber, J. (1993). From Tom Thumb to the Dockers: Some experiments with foraging robots. In Meyer, J., Roitblat, H. L., and Wilson, S. W., editors, *From Animals to Animats II*, pages 451–459. MIT Press, Cambridge, MA.

Ducatelle, F., Di Caro, G. A., Forster, A., Bonani, M., Dorigo, M., et al. (2014). Cooperative navigation in robotic swarms. *Swarm Intelligence*, 8:1–33.

Ducatelle, F., Di Caro, G. A., Pinciroli, C., and Gambardella, L. M. (2011). Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5(2):73–96.

Dugatkin, L. A., Mesterton-Gibbons, M., and Houston, A. I. (1992). Beyond the prisoners-dilemma-toward models to discriminate among mechanisms of cooperation in nature. *Trends in Ecology & Evolution*, (7):202–205.

Dussutour, A., Fourcassie, V., Helbing, D., and Deneubourg, J.-L. (2004). Optimal traffic organization in ants under crowded conditions. *Nature*, (428):70–73.

Eden, A. H. (2011). *Codecharts: Roadmaps and Blueprints for Object-Oriented Programs*. John Wiley and Sons, Hoboken, NJ.

Eden, A. H., Yehudai, A., and Gil, J. (1997). Precise specification and automatic application of design patterns. In *Proceedings 12th IEEE International Conference of Automated Software Engineering*, Piscataway, NJ. IEEE Press.

ESA Mars Missions (2015). Available on http://www.esa.int/Our_Activities/Space_Science/Mars_Express/Europe_reclaims_a_stake_in_Mars_exploration [Accessed on 04 Sep 2016].

ESA Space Applications Workshop (2014). Available on http://congrexprojects.com/2014-events/14c22/introduction [Accessed on 04 Sep 2016].

Fagen, R. (1987). A generalized habitat matching rule. *Evolutionary Ecology*, 1:5–10.

Farina, W. M., Grüter, C., and Díaz, P. C. (2005). Social learning of floral odours inside the honeybee hive. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 272:1923–1928.

Fernandez-Marquez, J. L., Di Marzo Serugendo, G., Montagna, S., Viroli, M., and Arcos, J. L. (2013). Description and composition of bio-inspired design patterns: A complete overview. *Natural Computing*, 12(1):43–67.

Ferrante, E. and Duéñez-Guzmán, E. (2013). GESwarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics. In Blum, C., Alba, E., Auger, et al., editors, *Proceedings of the Fifteenth International Conference on Genetic and Evolutionary Computation Conference Companion (GECCO 2013)*, pages 17–24, New York, NY. ACM.

Ferrante, E., Turgut, A. E., Dué, E., and Dorigo, M. (2015). Evolution of self-organized task specialization in robot swarms. *PLoSComputational Biology*, 11(8):e1004273.

Flemming, S. P., Smith, P. C., and Seymour, N. R. (1992). Short communications and commentaries ospreys use local enhancement and flock foraging to locate prey. *The Auk*, 3(109):649–654.

Fong, T., Bualat, M., Edwards, L., Fluckiger, L., Kuntz, C., et al. (2006). Human-robot site survey and sampling for space exploration. In *AIAA Space 2006*, Reston, VA. American Institute of Aeronautics and Astronautics.

Fong, T. and Nourbakhsh, I. (2000). Interaction challenges in human-robot space exploration. In Stone, W. C., editor, *Proceedings of the Fourth International Conference and Exposition on Robotics for Challenging Situations and Environments*, pages 340–346, Reston, VA. American Society of Civil Engineers.

Fraga, D., Gutiérrez, Á., Vallejo, J. C., Campo, A., and Bankovic, Z. (2011). Improving social odometry robot networks with distributed reputation systems for collaborative purposes. *Sensors (Basel, Switzerland)*, 11(12):11372–89.

Fretwell, S. D. (1972). *Populations in a Seasonal Environment.* Princeton University Press, Princeton, NJ.

Fujisawa, R., Dobata, S., Sugawara, K., and Matsuno, F. (2014). Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence*, 8(3):227–246.

Furniss, N. (1974). The Practical Significance of Decentralization. *The Journal of Politics*, 36(4):958–982.

Galef, B. G. and Wigmore, S. W. (1983). Transfer of information concerning distant foods: A laboratory investigation of the information-centre' hypothesis. *Animal Behaviour*, 31(3):748–758.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Pearson Education, Indianapolis, IN.

Gardelli, L., Viroli, M., and Omicini, A. (2007). Design patterns for self-organising systems. In Burkhard, H. D., Lindemann, G., Verbrugge, et al., editors, *Proceedings of the 5th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2007)*, pages 123–132, Berlin. Springer-Verlag.

Garnier, S., Tâche, F., Combe, M., Grimal, A., and Theraulaz, G. (2007). Alice in pheromone land: An experimental setup for the study of ant-like robots. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, pages 37–44, Piscataway, NJ. IEEE Press.

Gerkey, B. and Mataric, M. J. (2003). Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003)*, volume 3, pages 3862 – 3868, Piscataway, NJ. IEEE Press.

Giraldeau, L.-A., Valone, T. J., and Templeton, J. J. (2002). Potential disadvantages of using socially acquired information. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 357:1559–1566.

Gittleman, J. L. (1989). Carnivore group living: Comparative trends. In Gittleman, J. L., editor, *Carnivore Behavior, Ecology and Evolution*, pages 183–207. Ithaca, New York: Cornell University Press.

GNU Scientific Library (2001). The Levy alpha-stable distributions. Available on https://www.math.utah.edu/software/gsl/gsl-ref_278.html [Accessed on 04 Sep 2016].

Google Lunar X Prize (2015). Available on http://lunar.xprize.org [Accessed on 04 Sep 2016].

Grace, J. and Campora, C. (2005). Food location and discrimination by subterranean termites (*Isoptera: Rhinotermitidae*). In Lee, C.-Y. and Robinson, W. H., editors, *Proceedings of the Fifth International Conference on Urban Pests*, pages 437–441. P&Y Design Network, Malaysia.

Granovskiy, B., Latty, T., Duncan, M., Sumpter, D. J. T., and Beekman, M. (2012). How dancing honey bees keep track of changes: The role of inspector bees. *Behavioral Ecology*, 23(3):588–596.

Greenough, J. (2016). 10 million self-driving cars will be on the road by 2020. Business Insider UK. Available on http://uk.businessinsider.com/report-10-million-self-driving-cars-will-be-on-the-road-by-2020-2015-5-6 [Accessed on 04 Sep 2016].

Gregson, A. M., Hart, A. G., Holcombe, M., and Ratnieks, F. L. (2003). Partial nectar loads as a cause of multiple nectar transfer in the honey bee (*Apis mellifera*): A simulation model. *Journal of Theoretical Biology*, 222(1):1–8.

Griswold, A. (2016). Uber's self-driving cars are on the road. Quartz. Available on http://qz.com/688003/ubers-self-driving-cars-are-on-the-road/ [Accessed on 04 Sep 2016].

Groß, R., Nouyan, S., Bonani, M., Mondada, F., and Dorigo, M. (2008). Division of labour in self-organised groups. In Asada, M., Hallam, J. C. T., Meyer, J.-A., et al., editors, *Proceedings of the 10th International Conference on Simulation of Adaptive Behavior (SAB 2008)*, pages 426–436, Berlin. Springer.

Gutiérrez, Á., Campo, A., Monasterio-Huelin, F., Magdalena, L., and Dorigo, M. (2010). Collective decision-making based on social odometry. *Neural Computing and Applications*, 19(6):807–823.

Hamann, H. (2013). Towards swarm calculus: Urn models of collective decisions and universal properties of swarm performance. *Swarm Intelligence*, 7(2):145–172.

Hansenberger, S. and Wildhaber, I. (2016). Regulations on civilian drones in the us and europe, what do they involve? Robohub. Available on http://robohub.org/regulations-on-civilian-drones-in-the-us-and-europe-what-do-they-involve/ [Accessed on 04 Sep 2016].

Harel, D. and Rumpe, B. (2004). Meaningful modeling: what is the semantics of semantics? *Computer*, 37(10):64–72.

Hartley, R. V. L. (1928). Transmission of information. *Bell System Technical Journal*, pages 535–563.

Hayward, M. W. (2006). Prey preferences of the spotted hyaena (*Crocuta crocuta*) and degree of dietary overlap with the lion (*Panthera leo*). *Journal of Zoology*, 270(4):606–614.

Hecker, J. P., Letendre, K., Stolleis, K., Washington, D., and Moses, M. E. (2012). Formica ex machina: Ant swarm foraging from physical to virtual and back again. In Dorigo, M., Birattari, M., Blum, C., et al., editors, *Proceedings of the 8th International Conference on Swarm Intelligence (ANTS 2012)*, pages 252–259, Berlin. Springer.

Herrnstein, R. J. (1961). Relative and absolute strength of response as a function of frequency of reinforcement. *Journal of the experimental analysis of behavior*, 4:267–272.

Hoff, N., Sagoff, A., Wood, R. J., and Nagpal, R. (2010). Two foraging algorithms for robot swarms using only local communication. In *Proceedings of the 2010 IEEE International Conference on Robotics and Biomimetics (ROBIO 2010)*, pages 123–130, Piscataway, NJ. IEEE Press.

Hoff, N., Wood, R., and Nagpal, R. (2013). Distributed colony-level algorithm switching for robot swarm foraging. In Martinoli, A., Mondada, F., Correll, N., et al., editors, *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 417–430. Springer, Berlin.

Holekamp, K. E., Smith, J. E., Strelioff, C. C., Van Horn, R. C., and Watts, H. E. (2012). Society, demography and genetic structure in the spotted hyena. *Molecular ecology*, (21):613–32.

Hoogland, J. L. (1981). The evolution of coloniality in white-tailed and black-tailed prairie dogs. *Ecology*, 62(1):252–272.

Hrolenok, B., Luke, S., Sullivan, K., and Vo, C. (2010). Collaborative foraging using beacons. In van der Hoek, W., Kaminka, G. A., Lesperance, Y., et al., editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1197–1204, Richland, SC. IFAAMAS.

Huntsberger, T., Aghazarian, H., Baumgartner, E., and Schenker, P. S. (2000a). Behavior-based control systems for planetary autonomous robot outposts. In *Proceedings of the 2000 IEEE Aerospace Conference*, volume 7, pages 679 – 686, Piscataway, NJ. IEEE Press.

Huntsberger, T., Rodriguez, G., and Schenker, P. S. (2000b). Robotics challenges for robotic and human Mars exploration. In Stone, W. C., editor, *Proceedings of the Fourth International Conference and Exposition on Robotics for Challenging Situations and Environments (Robotics 2000)*, pages 340–346, Reston, VA. ASCE.

Jevtic, A. and Gazi, P. (2010). Building a swarm of robotic bees. In *World Automation Congress 2010*, pages 1–6, Piscataway, NJ. IEEE Press.

Jevtic, A., Gutiérrez, Á., Andina, D., and Jamshidi, M. (2012). Distributed bees algorithm for task allocation in swarm of robots. *IEEE Systems Journal*, 6(2):296–304.

Jones, C. and Mataric, M. J. (2003). Adaptive division of labor in large-scale minimalist multi-robot systems. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, pages 1969 – 1974, vol. 2, Piscataway, NJ. IEEE Press.

Kazama, T., Sugawara, K., and Watanabe, T. (2005). Traffic-like movement on a trail of interacting robots with virtual pheromone. In Murase, K., Sekiyama, K., Naniwa, T., et al., editors, *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, pages 383–388, Berlin. Springer.

Kendal, R. L. (2004). The role of conformity in foraging when personal and social information conflict. *Behavioral Ecology*, 15(2):269–277.

Kernbach, S., Habe, D., Kernbach, O., Thenius, R., Radspieler, G., et al. (2013). Adaptive collective decision-making in limited robot swarms without communication. *The International Journal of Robotics Research*, 32(1):35–55.

Kernbach, S., Nepomnyashchikh, V. a., Kancheva, T., and Kernbach, O. (2012). Specialization and generalization of robot behaviour in swarm energy foraging. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):131–152.

Krieger, M. J. B. and Billeter, J.-B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1-2):65–84.

Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2004). Efficiency and task allocation in prey retrieval. In Ijspeert, A. J., Murata, M., and Wakamiya, N., editors, *Biologically Inspired Approaches to Advanced Information Technology*, volume 3141 of *Lecture Notes in Computer Science*, pages 274–289. Springer, Berlin.

Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2006). Division of labour in a group of robots inspired by ants' foraging behaviour. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):4–25.

Lachlan, R., Crooks, L., and Laland, K. (1998). Who follows whom? Shoaling preferences and social learning of foraging information in guppies. *Animal behaviour*, 56(1):181–90.

Lee, J. H. and Ahn, C. W. (2011). Improving energy efficiency in cooperative foraging swarm robots using behavioral model. In Abdullah, R., Khader, A. T., Venkat, I., et al., editors, *Proceedings of the Sixth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA'11)*, pages 39–44, Piscataway, NJ. IEEE Press.

Lee, J. H., Ahn, C. W., and An, J. (2013). A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: An empirical analysis. In *Proceedings of the 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2013)*, pages 489–493, Piscataway, NJ. IEEE Press.

Lein, A. and Vaughan, R. T. (2009). Adapting to non-uniform resource distributions in robotic swarm foraging through work-site relocation. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, pages 601–606, Piscataway, NJ. IEEE Press.

Lemmens, N., de Jong, S., Tuyls, K., and Nowe, A. (2008). Bee behaviour in multi-agent systems. In Tuyls, K., Nowe, A., Guessoum, Z., et al., editors, *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, volume 4865 of *Lecture Notes in Computer Science*, pages 145–156. Springer, Berlin.

Lerman, K., Jones, C., Galstyan, A., and Mataric, M. J. (2006). Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotic Research*, 25:225–242.

Liu, W. and Winfield, A. F. T. (2010). Modelling and optimisation of adaptive foraging in swarm robotic systems. *The International Journal of Robotics Research*, 29(14):1743–1760.

Liu, W., Winfield, A. F. T., Sa, J., Chen, J., and Dou, L. (2007a). Strategies for energy optimisation in a swarm of foraging robots. In Sahin, E., Spears, W. M., and Winfield, A. F. T., editors, *Swarm Robotics*, volume 4433 of *Lecture Notes in Computer Science*, pages 14–26. Spinger, Berlin.

Liu, W., Winfield, A. F. T., Sa, J., Chen, J., and Dou, L. (2007b). Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, 15(3):289–305.

Martinelli, A. (2002). The odometry error of a mobile robot with a synchronous drive system. *IEEE Transactions on Robotics and Automation*, 18(3):399–405.

Martinoli, A., Easton, K., and Agassounon, W. (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research*, 23(4):415–436.

Mataric, M. J., Sukhatme, G. S., and Ostergaard, E. H. (2003). Multi-robot task allocation in uncertain environments. *Autonomous Robots*, 14(2-3):255–263.

Mather, T. W. and Hsieh, M. A. (2012). Ensemble modeling and control for congestion management in automated warehouses. In *Proceedings of the 2012 IEEE International Conference on Automation Science and Engineering (CASE 2012)*, pages 390–395, Piscataway, NJ. IEEE Press.

Mayet, R., Roberz, J., Schmickl, T., and Crailsheim, K. (2010). Antbots: A feasible visual emulation of pheromone trails for swarm robots. In Dorigo, M., Birattari, M., Di Caro, G. A., et al., editors, *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 84–94. Springer, Berlin.

McFarland, D. and Spier, E. (1997). Basic cycles, utility and opportunism in self-sufficient robots. *Robotics and Autonomous Systems*, 20:179–190.

Michelena, P., Jeanson, R., Deneubourg, J.-L., and Sibbald, A. M. (2010). Personality and collective decision-making in foraging herbivores. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 277(1684):1093–9.

Mikkonen, T. (1998). Formalizing design patterns. In *Proceedings of the 20th International Conference on Software Engineering (ICSE'98)*, pages 115–124, Piscataway, NJ. IEEE Press.

Miletitch, R., Trianni, V., Campo, A., and Dorigo, M. (2013). Information aggregation mechanisms in social odometry. In Liò, P., Miglino, O., Nicosia, G., et al., editors,

Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems (ECAL 2013), pages 102–109, Cambridge, MA. MIT Press.

Miller, J. M., Wang, X. R., Lizier, J. T., Prokopenko, M., and Rossi, L. F. (2014). Measuring information dynamics in swarms. In Prokopenko, M., editor, *Guided Self-Organisation: Inception*, volume 9 of *Emergence, Complexity and Computation*, pages 343–364. Springer, Berlin.

Möbius, Y., Boesch, C., Koops, K., Matsuzawa, T., and Humle, T. (2008). Cultural differences in army ant predation by West African chimpanzees? A comparative study of microecological variables. *Animal Behaviour*, 76(1):37–45.

Moussaid, M., Garnier, S., Theraulaz, G., and Helbing, D. (2009). Collective information processing and pattern formation in swarms, flocks, and crowds. *Topics in Cognitive Science*, 1(3):469–497.

Nagpal, R. (2004). A catalog of biologically-inspired primitives for engineering self-organization. In Di Marzo Serugendo, G., Karageorgos, A., Rana, O. F., et al., editors, *Engineering Self-Organising Systems*, pages 53–62. Springer, Berlin.

NASA Mars Missions (2015). Available on http://mars.nasa.gov/programmissions/missions/future/ [Accessed on 04 Sep 2016].

Newton-Fisher, N. E. (2007). Chimpanzee hunting behavior. In Henke, W. and Tattersall, I., editors, *Handbook of Paleoanthropology. Volume 2. Primate Evolution and Human Origins*, pages 1295–1320. Springer, Berlin.

Niv, Y., Joel, D., Meilijson, I., and Ruppin, E. (2002). Evolution of reinforcement learning in uncertain environments: A simple explanation for complex foraging behaviors. *Adaptive Behavior*, 10(1):5–24.

Noble, J. and Todd, P. M. (2002). Imitation or something simpler? Modeling simple mechanisms for social information processing. In Dautenhahn, K. and Nehaniv, C., editors, *Imitation in Animals and Artifacts*, pages 423–439. MIT Press, Cambridge, MA.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2009). Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711.

Nutonomy (2016). Available on http://www.nutonomy.com [Accessed on 04 Sep 2016].

Object Management Group (2015). Unified Modeling Language specification. Available on http://www.omg.org/spec/UML/ [Accessed on 04 Sep 2016].

Pais, D., Caicedo-Núnez, C. H., and Leonard, N. E. (2012). Hopf bifurcations and limit cycles in evolutionary network dynamics. *SIAM Journal on Applied Dynamical Systems*, 11(4):1754–1784.

Parker, L. E. (1995). The effect of action recognition and robot awareness in cooperative robotic teams. In *Proceedings of the 1995 IEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1995)*, pages 212 – 219, Piscataway, NJ. IEEE Press.

Parunak, H. and Brueckner, S. A. (2004). Engineering swarmings systems. In Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors, *Methodologies and Software Engineering for Agent Systems*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 341–376. Springer, Berlin.

Parunak, H. V. D. and Brueckner, S. A. (2015). Software engineering for self-organizing systems. *The Knowledge Engineering Review*, 30(4):419–434.

Paull, L., Seto, M., and Leonard, J. J. (2014). Decentralized cooperative trajectory estimation for autonomous underwater vehicles. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 184–191, Piscataway, NJ. IEEE Press.

Pérez-Uribe, A. (2001). Using a time-delay actor-critic neural architecture with dopamine-like reinforcement signal for learning in autonomous robots. In Wermter, S., Austin, J., and Willshaw, D., editors, *Emergent Neural Computational Architectures Based on Neuroscience*, volume 2036 of *Lecture Notes in Computer Science*, pages 522–533. Springer, Berlin.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., et al. (2012). ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295.

Pini, G., Brutschy, A., Pinciroli, C., Dorigo, M., and Birattari, M. (2013). Autonomous task partitioning in robot foraging: An approach based on cost estimation. *Adaptive Behavior*, 21(2):118–136.

Pitonakova, L., Crowder, R., and Bullock, S. (2014). Understanding the role of recruitment in collective robot foraging. In Lipson, H., Sayama, H., Rieffel, J., et al., editors, *Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*, pages 264–271, Cambridge, MA. MIT Press.

Pitonakova, L., Crowder, R., and Bullock, S. (2016a). Information flow principles for plasticity in foraging robot swarms. *Swarm Intelligence*, 10(1):33–63.

Pitonakova, L., Crowder, R., and Bullock, S. (2016b). Task allocation in foraging robot swarms: The role of information sharing. In Gershenson, G., Froese, T., Siqueiros, J. M., et al., editors, *Proceedings of the Fifteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE XV)*, pages 306–313, Cambridge, MA. MIT Press.

Polani, D., Prokopenko, M., and Yaeger, L. S. (2013). Information and self-organization of behavior. *Advances in Complex Systems*, 16(2 & 3).

Prabhakar, B., Dektar, K. N., and Gordon, D. M. (2012). The regulation of ant colony foraging activity without spatial information. *PLoS computational biology*, 8(8):e1002670.

Raman, V. (2014). Reactive switching protocols for multi-robot high-level tasks. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 336 – 341, Piscataway, NJ. IEEE Press.

Ranjbar-Sahraei, B., Weiss, G., and Nakisaei, A. (2012). A multi-robot coverage approach based on stigmergic communication. In Timm, I. J. and Guttmann, C., editors, *Multiagent System Technologies*, volume 7598 of *Lecture Notes in Computer Science*, pages 126–138. Springer, Berlin.

Ranta, E., Rita, H., and Lindstrom, K. (1993). Competition versus cooperation: Success of individuals foraging alone and in groups. *The American Naturalist*, 142(1):42–58.

Ratnieks, F. L. W. (2008). Biomimicry: Further insights from ant colonies? In Lio, P., Yoneki, E., Crowcroft, J., et al., editors, *Bio-Inspired Computing and Communication*, pages 58–66. Spinger-Verlag, Berlin.

Reina, A., Dorigo, M., and Trianni, V. (2014). Towards a cognitive design pattern for collective decision-making. In Hutchison, D., Kanade, T., Kittler, J., et al., editors, *Swarm Intelligence*, volume 8667 of *Lecture Notes in Computer Science*, pages 194–205. Springer, Berlin.

Reina, A., Valentini, G., Fernández-oto, C., Dorigo, M., and Trianni, V. (2015). A Design Pattern for Decentralised Decision Making. *PloS One*, 10(10):e0140950.

Reynolds, A. M. and Rhodes, C. J. (2009). The Lévy flight paradigm: Random search patterns and mechanisms. *Ecology*, 90(4):877–887.

Ribeiro, P. L., Helene, A. F., Xavier, G., Navas, C., and Ribeiro, F. L. (2009). Ants can learn to forage on one-way trails. *PloS one*, 4(4):e5024.

Robinson, E. J. H., Jackson, D. E., Holcombe, M., and Ratnieks, F. L. W. (2005). Insect communication: 'No entry' signal in ant foraging. *Nature*, 438(7067):442.

Robinson, S. K. and Holmes, R. T. (1982). Foraging behaviour of forest birds: The relationships among search tactics, diet and habitat structure. *Ecology*, 63(6):1918–1931.

Russell, R. A. (1999). Ant trails - an example for robots to follow? In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pages 2698 – 2703 vol. 4, Piscataway, NJ. IEEE Press.

Rybski, P. E., Larson, A., Veeraraghavan, H., Lapoint, M., and Gini, M. (2007). Communication strategies in multi-robot search and retrieval: Experiences with MinDART. In Alami, R., Chatila, R., and Asama, H., editors, *Distributed Autonomous Robotic Systems 6*, pages 317–326. Springer, Berlin.

Sanburn, J. (2012). Self-driving cars available by 2019, report says. Time. Available on http://business.time.com/2012/08/16/self-driving-cars-available-by-2019-report-says/ [Accessed on 04 Sep 2016].

Sarker, M. O. F. and Dahl, T. S. (2011). Bio-Inspired communication for self-regulated multi-robot systems. In Yasuda, T., editor, *Multi-Robot Systems, Trends and Development*, pages 367–392. InTech, DOI: 10.5772/13104.

Schenker, P. S., Huntsberger, T. L., Pirjanian, P., Baumgartner, E., Aghazarian, H., et al. (2001). Robotic automation for space: planetary surface exploration, terrain-adaptive mobility, and multi-robot cooperative tasks. In Casasent, D. P. and Hall, E., editors, *Proceedings of SPIE Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision*, pages 12–28, Washington, DC. SPIE.

Schmickl, T. and Crailsheim, K. (2008). Trophallaxis within a robotic swarm: Bio-inspired communication among robots in a swarm. *Autonomous Robots*, 25(1):171–188.

Schmickl, T. and Hamann, H. (2010). BEECLUST: A swarm algorithm derived from honeybees. In Xiao, Y., editor, *Bio-inspired Computing and Networking*, pages 95–137. CRC Press, Boca Raton, FL.

Schmickl, T., Möslinger, C., and Crailsheim, K. (2007). Collective perception in a robot swarm. In Sahin, E., Spears, W. M., and Winfield, A. F. T., editors, *Swarm Robotics*, volume 4433 of *Lecture Notes in Computer Science*, pages 144–157. Springer, Berlin.

Schmickl, T., Thenius, R., and Crailsheim, K. (2012). Swarm-intelligent foraging in honeybees: Benefits and costs of task-partitioning and environmental fluctuations. *Neural Computing and Applications*, 21(2):251–268.

Schmidt, D. C. (1995). Using design patterns to develop reuseable object-oriented communication software. *Communications of the ACM Special Issue on Object-Oriented Experiences*, 38(10):65–74.

Seeley, T. D. (1992). The tremble dance of the honey bee: Message and meanings. *Behavioral Ecology and Sociobiology*, 31(6):375–383.

Seeley, T. D. (1994). Honey bee foragers as sensory units of their colonies. *Behavioral Ecology and Sociobiology*, 34(1):51–62.

Seeley, T. D., Camazine, S., and Sneyd, J. (1991). Collective decision-making in honey bees: How colonles choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28:277–290.

Seeley, T. D. and Morse, R. a. (1976). The nest of the honey bee (*Apis mellifera L.*). *Insectes Sociaux*, 23:495–512.

Seeley, T. D., Visscher, P. K., Schlegel, T., Hogan, P. M., Franks, N. R., et al. (2012). Stop signals provide cross inhibition in collective decision-making by honeybee swarms. *Science*, 335(6064):108–111.

Serugendo, G. M., Gleizes, M.-P., and Karageorgos, A. (2006). Self-organisation and emergence in MAS: An overview self-organisation. *Informatica*, 30:45–54.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

Shell, D. A. and Mataric, M. J. (2006). On foraging strategies for large-scale multi-robot systems. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, pages 2717 – 2723, Piscataway, NJ. IEEE Press.

Skinner, J. D., Van Aarde, R. J., and Goss, R. A. (1995). Space and resource use by brown hyenas *Hyaena hrullnea* in the Namib Desert. *Journal of Zoology London*, (237):123–131.

Smith, E. A. (2010). Communication and collective action: Language and the evolution of human cooperation. *Evolution and Human Behavior*, 31(4):231–245.

Sperati, V., Trianni, V., and Nolfi, S. (2011). Self-organised path formation in a swarm of robots. *Swarm Intelligence*, 5(2):97–119.

Spier, E. and McFarland, D. (1997). Possibly optimal decision-making under self-sufficiency and autonomy. *Journal of theoretical biology*, 189(3):317–31.

Sridhar, H., Beauchamp, G., and Shanker, K. (2009). Why do birds participate in mixed-species foraging flocks? A large-scale synthesis. *Animal Behaviour*, 78(2):337–347.

Stander, P. E. and Albon, S. D. (1993). Hunting success of lions in a semi-arid environment. *Symposia of the Zoological Society of London*, (65):127–143.

Starship Technologies (2016). Press release. Available on https://www.starship.xyz/starship-technologies-launches-testing-program-self-driving-delivery-robots-major-industry-partners/ [Accessed on 04 Sep 2016].

Stiefelhagen, M., Van Der Werff, K., Meijer, B. R., and Tomiyama, T. (2004). Distributed autonomous agents, navigation and cooperation with minimum intelligence in a dynamic warehouse application. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 6, pages 5573–5578, Piscataway, NJ. IEEE Press.

Sugawara, K. and Watanabe, T. (2002). Swarming robots - foraging behavior of simple multirobot system. In *Proceedings of the 2002 IEEE/RSI Internatinal Conference on Intelligent Robots and Systems (IROS 2002)*, pages 2702–2707, Piscataway, NJ. IEEE Press.

Sumpter, D. J. T. and Beekman, M. (2003). From nonlinearity to optimality: Pheromone trail foraging by ants. *Animal Behaviour*, 66(2):273–280.

Tahara, Y., Ohsuga, A., and Honiden, S. (1999). Agent system development method based on agent patterns. In *Proceedings of the 20th International Conference on Software Engineering (ICSE'99)*, pages 356–367, Piscataway, NJ. IEEE Press.

Tereshko, V. and Loengarov, A. (2005). Collective decision-making in honey bee foraging dynamics. *Computing and Information Systems Journal*, 9(3):1–7.

Thenius, R., Schmickl, T., and Crailsheim, K. (2008). Optimisation of a honeybee-colony's energetics via social learning based on queuing delays. *Connection Science*, 20(2-3):193–210.

Thiel, S., Häbe, D., and Block, M. (2009). Co-operative robot teams in a hospital environment. In *Proceedings of the 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 2009)*, volume 2, pages 843–847, Piscataway, NJ. IEEE Press.

Trianni, V., Tuci, E., Passino, K. M., and Marshall, J. A. R. (2011). Swarm cognition: An interdisciplinary approach to the study of self-organising biological collectives. *Swarm Intelligence*, 5:3–18.

Tukey, J. (1949). Comparing individual means in the analysis of variance. *Biometrics*, 5(2):99–114.

Ulam, P. and Balch, T. (2004). Using optimal foraging models to evaluate learned robotic foraging behavior. *Adaptive Behavior*, 12(4):213–222.

Valdastri, P., Corradi, P., Menciassi, A., Schmickl, T., Crailsheim, K., et al. (2006). Micromanipulation, communication and swarm intelligence issues in a swarm microrobotic platform. *Robotics and Autonomous Systems*, 54(10):789–804.

Valentini, G., Hamann, H., and Dorigo, M. (2014). Self-organized collective decision making: The weighted voter model. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 45–52, New York. ACM.

Vane, G., Goswami, J. N., Zelenyi, L., and Foing, B. (2010). *Future Planetary Robotic Exploration: The Need for International Cooperation*. International Academy of Astronautics, Paris. Available on http://shop.iaaweb.org/?q=node/434 [Accessed on 04 Sep 2016].

Vivaldini, K., Galdames, J. P. M., Pasqual, T. B., Sobral, R. M., Araujo, R. C., et al. (2010). Automatic routing system for intelligent warehouses. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA 2010)*, pages 93–98, Piscataway, NJ. IEEE Press.

von Frisch, K. (1967). *The dance language and orientation of bees*. Harvard University Press, Cambridge, MA.

Wang, X. R., Miller, J. M., Lizier, J. T., Prokopenko, M., and Rossi, L. F. (2012). Quantifying and tracing information cascades in swarms. *PloS one*, 7(7):e40084.

Ward, A. J. W., Krause, J., and Sumpter, D. J. T. (2012). Quorum decision-making in foraging fish shoals. *PloS one*, 7(3):e32411.

Ward, P. and Zahavi, A. (1973). The importance of certain assemblages of birds as "information-centers" for food-finding. *Ibis*, (115):517–534.

Wawerla, J. and Vaughan, R. T. (2010). A fast and frugal method for team-task allocation in a multi-robot transportation system. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA 2010)*, pages 1432–1437, Piscataway, NJ. IEEE Press.

Webster, M. M. and Laland, K. N. (2012). Social information, conformity and the opportunity costs paid by foraging fish. *Behavioral Ecology and Sociobiology*, 66(5):797–809.

Weimerskirch, H., Le Corre, M., Jaquemet, S., Potier, M., and Marsac, F. (2004). Foraging strategy of a top predator in tropical waters: Great frigatebirds in the Mozambique Channel. *Marine Ecology Progress Series*, 275:297–308.

Winfield, A. F. T. (2009). Towards an engineering science of robot foraging. In Asama, H., Kurokawa, H., and Sekiyama, K., editors, *Distributed Autonomous Robotic Systems 8*, pages 185–192. Springer, Berlin.

Wray, M. K., Klein, B. a., and Seeley, T. D. (2011). Honey bees use social information in waggle dances more fully when foraging errors are more costly. *Behavioral Ecology*, 23(1):125–131.

Yang, Y., Zhou, C., and Tian, Y. (2009). Swarm robots task allocation based on response threshold model. In *Proceedings of the 4th International Conference on Autonomous Robots and Agents (ICARA 2009)*, pages 171–176, Piscataway, NJ. IEEE Press.

Yasuda, T., Kage, K., and Ohkura, K. (2014). Response threshold-based task allocation in a reinforcement learning robotic swarm. In *Proceedings of the 2014 IEEE 7th International Workshop on Computational Intelligence and Applications (IWCIA 2014)*, pages 189–194, Piscataway, NJ. IEEE Press.

Zahadat, P., Crailsheim, K., and Schmickl, T. (2013). Social inhibition manages division of labour in artificial swarm systems. In Liò, P., Miglino, O., Nicosia, G., et al., editors, *Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems (ECAL 2013)*, pages 609–616, Cambridge, MA. MIT Press.

Zhang, D., Xie, G., Yu, J., and Wang, L. (2007). Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*, 55(7):572–588.

Ziegler, C. (2015). Google will start a new company for its self-driving car, and it might take on uber. The Verge. Available on http://www.theverge.com/2015/12/16/10288104/google-alphabet-self-driving-car-ride-hailing-uber [Accessed on 04 Sep 2016].