UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
Electronics and Computer Science

Resource Allocation Methods for Fog Computing Systems

by

Fan Bi
ORCiD: 0000-0003-1844-9685

A thesis for the degree of
Doctor of Philosophy

December 2022

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
Electronics and Computer Science

Doctor of Philosophy

Resource Allocation Methods for Fog Computing Systems

by Fan Bi

Fog computing is gaining popularity as a suitable computer paradigm for the Internet of things (IoT). It is a virtualised platform that sits between IoT devices and centralised cloud computing. Fog computing has several characteristics, including proximity to IoT devices, low latency, geo-distribution, a large number of fog nodes, and real-time interaction. A key challenge in fog is resource allocation because existing resource allocation methods for cloud computing cannot directly apply to fog computing. Hence, many resource allocation methods for fog computing have been proposed since the birth of fog computing. However, most of these methods are centralised and not truthful, which means that users are not incentivised always to provide the true information of their tasks and their efficiency could decrease significantly if some users are strategic. Hence, an efficient resource allocation mechanism for this computing paradigm, which can be used in a strategic environment, is in need. Furthermore, a decentralised resource allocation algorithm is needed when there is no central control in the fog computing system. To this purpose, we consider three challenges: (1) near-optimal resource allocation in a fog system; (2) incentivising self-interested IoT users to truthfully report their tasks; and (3) decentralised resource allocation in a fog system.

In this thesis, we examine relevant literature and describe its achievements and shortcomings. Currently, many resource allocation mechanisms using various techniques are proposed for resource allocation in cloud computing and fog computing. However, there is little work that studies truthful fog computing resource allocation mechanisms. Furthermore, reinforcement learning is also widely used in resource allocation for fog computing. However, most of these studies focus on single-agent reinforcement learning and centralised resource allocation. In summary, they only address a subset of the challenges in our fog computing resource allocation problem, and their application scenarios are highly limited.

Therefore, we introduce our resource allocation model, i.e., Resource Allocation in Fog Computing (RAFC) and Distributed Resource Allocation in Fog Computing (DRAFC) in detail and choose the benchmark mechanisms to evaluate our proposed resource allocation mechanisms. Then, we develop and test an efficient and truthful mechanism called Flexible Online Greedy (FlexOG) using simulations. The simulations demonstrate that our mechanism can reach a higher level of social welfare than the truthful benchmark mechanisms by up to 10% and that it often achieves about 90% of the theoretical upper bound. To make FlexOG more scalable, we propose a modification of FlexOG called Semi-FlexOG, which is shown to use less processing time. Furthermore, to allocate resources in a decentralised fog system, we propose Decentralised Auction with PPO (DAPPO), which uses online reverse auctions and decentralised reinforcement learning for allocating tasks to resources in the fog. By enabling competition between resource providers, these auctions ensure that the most suitable provider is chosen for a given task, but without the computational and communication overheads of a centralised solution. In order to derive effective bidding strategies for nodes, we use a Proximal Policy Optimisation (PPO) reinforcement learning algorithm that takes into account the status of a node and task characteristics and that aims to maximise the node's long-term revenue. Hence, DAPPO deals naturally with highly dynamic systems, where the pattern of tasks could change dramatically. The results of our simulations show that DAPPO achieves a good performance in terms of social welfare. Specifically, its performance is close to the upper bound (around 90%) and better than benchmarks (0% to 30%). Finally, we conclude and outline possible future work.

Contents

List of Figures

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. 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;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. 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;
5. I have acknowledged all main sources of help;
6. 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;
7. Parts of this work have been published as specifiec in Section 1.4

Signed: Date:

Acknowledgements

I would like to thank my wife, parents, co-authors, colleagues, friends, supervisors Sebastian Stein, Enrico Gerding and Nick Jennings, progression report examiners, and the PhD funding provider (The International Technology Alliance in Distributed Analytics and Information Sciences[3]) for their accompany, guidance and support during my PhD journey.

[3]https://dais-ita.org/pub

Acronyms

AI Artificial Intelligence.

AR Augmented Reality.

AV Automonous Vehicle.

AWS Amazon Web Services.

DAPPO Decentralised Auction with PPO.

Dec-POMDP Decentralised Partially Observable Markov Decision Process.

DQN Deep Q Network.

DRAFC Distributed Resource Allocation in Fog Computing.

DSIC Dominant-Strategy Incentive Compatible.

EC2 Elastic Compute Cloud.

EV Electric Vehicle.

FlexOG Flexible Online Greedy.

IoT Internet of things.

ISP Iternet Service Provider.

MARL Multi-Agent Reinforcement Learning.

MDP Markov Decision Processe.

MDPs Markov Decision Processes.

OG Online Greedy.

POMDP Partially Observable Markov Decision Processe.

PPO Proximal Policy Optimisation.

RACC Resource Allocation in Cloud Computing.

RAFC Resource Allocation in Fog Computing.

TD Temporal Difference.

VM Virtual Machine.

VR Virtual Reality.

Chapter 1

Introduction

To extend the traditional Internet, which mainly connects computers and smartphones, the IoT is about connecting all kinds of physical devices in the world to the Internet, and we call those things IoT devices in this thesis. The IoT is developing rapidly — by 2021, the number of connected IoT devices had grown to 12.3 billion, and it is estimated that by 2025, there will be 27 billion active devices in the IoT (Sinha, 2021). The reason why the IoT is fast-developing is that it makes things smart by giving them the ability to receive and send information to the Internet. IoT applications such as smart homes (Ricquebourg et al., 2006; Stolojescu-Crisan et al., 2021), smart cities (Cocchia, 2014; Hassan et al., 2021), smart agriculture (TongKe, 2013; Sinha and Dhanalakshmi, 2022), smart grid (Tuballa and Abundo, 2016; Mehmood et al., 2021), and industry 4.0 (Roblek et al., 2016; Khan and Javaid, 2021) can significantly improve work efficiency, make life more convenient and improve our health. For example, a smart city can have a traffic management system, which takes advantage of widely distributed sensors such as video cameras, Bluetooth sensors (i.e., sensors that detect how many smartphones with their Bluetooth on are near it.), and loop detectors (detect vehicles arriving or passing it) to reduces traffic congestion (Su et al., 2011), and a smart home may allow a refrigerator to buy food automatically according to its stock (Ricquebourg et al., 2006).

However, IoT devices usually have very limited computing power because they need to have extremely low cost and very low power consumption (Chen et al., 2014). For instance, to reduce cost, many IoT devices only use energy harvested from the environment (Liu and Ansari, 2019), which is very limited. Many IoT devices must be low-cost so that people are willing to buy and use them. For this reason, many IoT devices cannot process their computational tasks or store their data locally. For example, as a surveillance camera in a traffic management system generates a vast amount of video data every day, its limited storage will fill up quickly if it stores the data locally. Another example is the Amazon Echo, a smart speaker developed by Amazon that performs speech recognition by sending the voice to Amazon's servers because the Echo does not have enough computational power to perform this locally (Marr, 2018). Similarly, the data generated by weather sensors in smart agriculture need to be gathered for analysis because weather sensors can only collect and send weather data, and the aggregated data from many sensors is required in order to predict the weather (Mekala and Viswanathan, 2017).

One way to solve this problem is by combining IoT with cloud computing (Chen et al., 2014; Sadeeq et al., 2021). This is because the cloud can offer virtually unlimited computational power and storage from its data centres to IoT devices (Botta et al., 2016). In more detail, the data centre in the cloud is mainly a collection of computing facilities (e.g., servers, routers and switches) that are used to create cloud services (Greenberg et al., 2008). However, this approach also has three main deficiencies in some IoT scenarios. Firstly, cloud computing often has high latency, and thus, latency-sensitive applications are not feasible to be deployed in the cloud (Bonomi et al., 2012). For instance, a study of two cloud gaming platforms shows that the latency for their games is between 135-500ms (Chen et al., 2011), which is too high for some types of games such as first-person shooters and multiplayer online battle arena games. Secondly, the bandwidth to and from the cloud provider is a bottleneck of cloud computing, especially when more and more IoT devices will connect to the Internet in the future (Sarkar et al., 2018). Finally, the data privacy and security of cloud computing faces many risks (Takabi et al., 2010). For example, many IoT devices do not have the capability to encrypt their data before sending it to the cloud (Alrawais et al., 2017). Hence, cloud computing is not suitable for many IoT applications that are latency-sensitive, generate a large volume of traffic or require a high-security level (e.g., smart homes, smart cities and smart grid) (Sen, 2015).

Thus, we need to turn to a new computing paradigm to make up for these deficiencies of cloud computing. Against this background, fog computing is proposed as a promising complement to cloud computing for IoT applications (Bonomi et al., 2012). Fog computing is similar to cloud computing that provides computing, networking and storage services between IoT devices and the cloud (Bonomi et al., 2012). Fog computing reduces latency and saves network bandwidth requirements by processing data close to IoT devices. Furthermore, fog computing makes IoT systems more secure. For instance, it reduces the chances of eavesdropping by processing data locally and reducing the amount of sensitive data transmitted to the cloud (Bonomi et al., 2012). For example, fog computing can process the video stream collected from a video surveillance system, which is latency-sensitive, privacy-sensitive and generates a log of data to track objects (Liu et al., 2018). A new secure data storage and searching framework, where raw data are first processed and stored in the fog, and then non-time-sensitive data are sent to the cloud, is proposed and verified to significantly improve the data security in the IoT because the amount of data transmitted in the network is decreased (Fu et al., 2018).

Besides the benefits of fog computing mentioned above, the effectiveness of fog is very much determined by its resource allocation mechanisms (see Chapter 1.2), and resource allocation is critical to making full use of resources in the fog and improving the Quality of Service (QoS) of the fog (Mahmud et al., 2018). Therefore, in this thesis, we focus on RAFC. In particular, we study RAFC in a strategic setting (i.e., assuming IoT users are rational) due to the fact that the information of IoT users' tasks is usually private, and IoT users are not necessarily truthful in reality. Furthermore, we also study DRAFC where there is no central control responsible for resource allocation, which is a common situation.

Note that there are also many challenges faced by fog computing. First, the fog computing system can be more complex and challenging to manage because it has many fog nodes at different locations. Second, regarding security and privacy, some fog nodes may be in a less secure environment. For example, a hacker may fake its IP address to gain access to a particular fog node (Rafiq et al., 2019). Third, authentication is more difficult for fog computing (Munir and Mohammed, 2019). For example, an illegal fog node may pretend to be legal and takes advantage of its users. Finally, regarding energy consumption, it takes a lot of energy to make all fog nodes work in a fog (Mukherjee et al., 2019).

In the following, we introduce the fog computing paradigm in more detail in Sections 1.1 and 1.2 and present the research challenges of RAFC in Section 1.3. Moreover, we highlight our research contributions and give a brief outline of the whole thesis in Sections 1.4 and 1.5 respectively.

1.1 Fog Computing Overview

Fog computing is formally defined as a virtualised[1] computing platform, which lies between IoT devices and cloud computing data centres, providing storage, computing and networking services, which is typically located at the edge of the network (i.e., the periphery of the network) (Bonomi et al., 2012). In particular, the main components of the fog are fog nodes, which are devices or facilities that can provide computing resources to IoT devices at the edge of the network, such as routers, switches, industrial controllers, smartphones, laptops, video surveillance cameras and base stations (Yi et al., 2015). Some commercial offerings of fog nodes are illustrated in Figure 1.1.

Compared with centralised data centres of the cloud, fog nodes have limited computational power and are distributed, heterogeneous, and closer to IoT devices (Tordera et al., 2016). For instance, Google data centres, which provide services

[1]Virtualisation is a technique to create virtual versions of resources such as an operating system or a storage device. It allows many users to share a single physical server and different operating systems, and applications can run on the same hardware simultaneously.

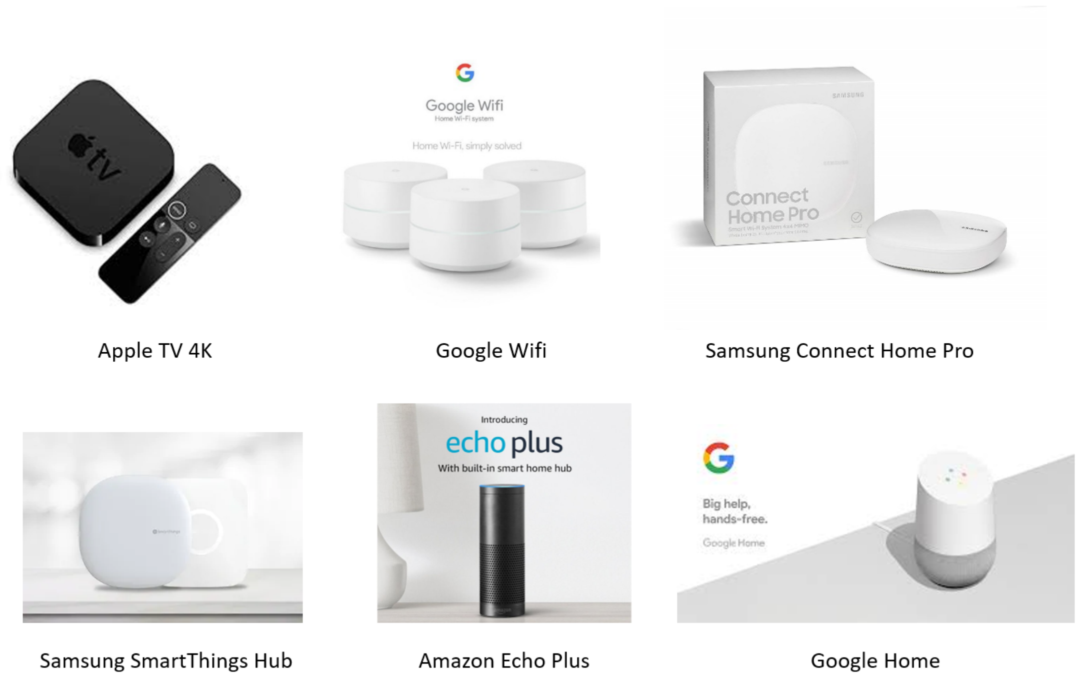

FIGURE 1.1: Some existing products that can serve as fog nodes.

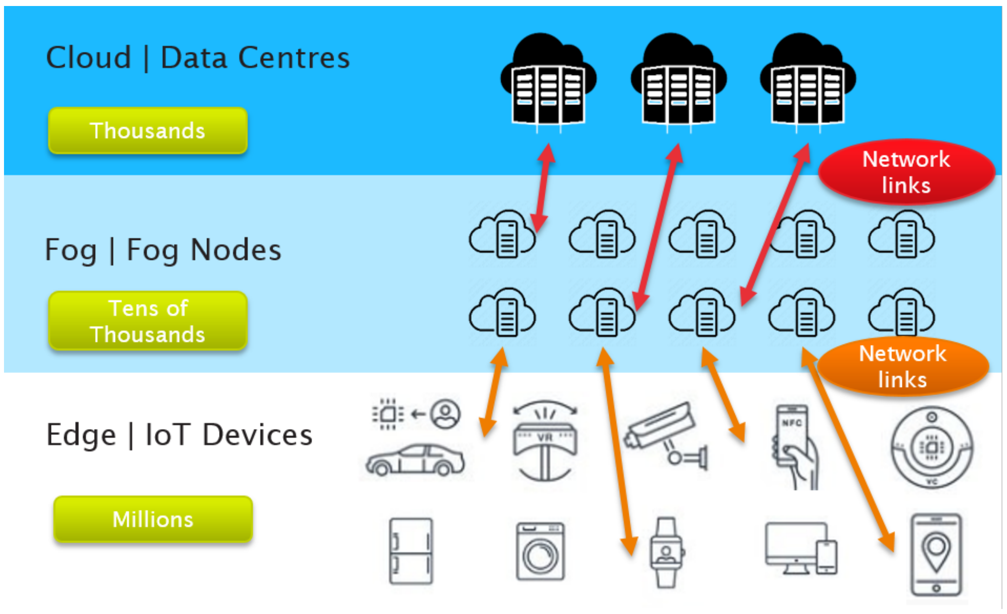

FIGURE 1.2: The architecture of the IoT, fog computing and cloud computing[3].

for Google users, are only deployed at 23 locations around the globe[2], and only two of them are in Asia. In contrast, many fog nodes would need to be deployed just along a motorway to provide low-latency services for Automonous Vehicle (AV)s (Xiao and Zhu, 2017). Figure 1.2 shows the architecture of the IoT and the position of fog computing in this architecture (Bonomi et al., 2012).

[2]https://www.google.com/about/datacenters/inside/locations/index.html

Note that several computing paradigms are similar to fog computing, such as geo-distributed clouds (Narayanan et al., 2014; Li et al., 2021), edge computing (Garcia Lopez et al., 2015; Al-Ansi et al., 2021), mobile cloud computing (Fernando et al., 2013; AlAhmad et al., 2021) and cloudlets (Satyanarayanan et al., 2009; Mukherjee et al., 2021). Although there are many differences between these computing paradigms, in essence, they all place computational resources close to the users, and they all have similar resource allocation models. As the name suggests, a geo-distributed cloud consists of several geo-distributed cloud data centres, which offer many advantages such as low latency, the ability to safeguard against failures, and exploitation of different regional energy prices. However, compared with fog nodes, the data centres in a geo-distributed cloud are still far away from IoT devices (Narayanan et al., 2014). Furthermore, edge computing pushes the computing power directly into the edge of the Internet, that is, applications are processed on IoT devices that have sufficient computing power (Garcia Lopez et al., 2015), and mobile cloud computing is about offloading computing tasks from mobile devices to computing resource providers such as cloud or fog (Fernando et al., 2013). In addition, cloudlets is just an alternative name for fog (Satyanarayanan et al., 2009). Given this, in the rest of the thesis, we only refer to fog computing because it is the most appropriate platform for many IoT applications (Bonomi et al., 2012). Still, our discussion and solutions could be partly applicable to the other types of systems mentioned.

Similar to the cloud, the primary function of the fog is to provide computing and networking resources, and fog computing is rather a complement to cloud computing than a substitute (Matt, 2018). The differences between them make fog computing more suitable for many IoT application scenarios. In what follows, we highlight important features that fog computing should have (Bonomi et al., 2012), which need to be considered when formulating the resource allocation model for RAFC. Note that fog computing was recently proposed and has not yet been widely used, and some of these features are shared by both fog and cloud computing.

First, the fog should have the ability to execute low latency computational tasks for applications such as AVs, real-time video analytics and online games. For example, at high speed, the response time for the autopilot system of an AV to avoid an accident can be just several milliseconds. Fog computing can reduce latency and network traffic because the data collected by IoT devices will be sent to fog nodes nearby for processing and storage instead of sending them to often far-away cloud data centres (Atlam et al., 2018).

Second, since many IoT devices are mobile (e.g., smartphones, laptops, AVs), the fog

[3]Source of icons: `https://www.shutterstock.com/image-illustration/set-line-icons-open-path-internet-600405563`

should support mobility through wireless access. It can use techniques like Locator ID Separation Protocol[4], Mobile IPv6 and Hierarchical Mobile IPv6 (Liu et al., 2015). For example, LISP separates the address space of locators and identifiers and has the advantage of mobility. In addition, the mobility of IoT devices brings new challenges to resource allocation, such as VM migration and dynamic traffic routing.

Third, the fog also needs to support real-time interactions besides batch processing because real-time interactions are demanded by many IoT applications such as Virtual Reality (VR) and Augmented Reality (AR).

Finally, some data of the IoT needs to be processed in the fog, and others are more suitable to be processed in the cloud because the cloud has data that the fog does not have or has a greater computing power (Bonomi et al., 2012). As a result, the fog should also support interplay with the cloud, which means that the fog can send data to the cloud to process. For example, the big data generated by a smart grid can be first processed in the fog, and then the results are sent to the cloud for latency-insensitive analysis (Borylo et al., 2016).

Next, we will introduce the application scenarios that are especially suitable for fog computing, such as AR, VR, real-time video analytics, smart grid and AV. These applications typically need low latency that cannot be fulfilled by cloud computing.

- **AR and VR:** AR can "augment" real-world scenes by adding computer-generated graphics to the real world, and VR constructs an environment of computer-generated feedback of video and sound that makes people have the feeling that the generated world is immersive and realistic. Examples of AR include Microsoft HoloLens (Evans et al., 2017) and Google Glass (Al-Maroof et al., 2021), while HTC Vive (Dempsey, 2016) and Oculus Rift (Jost et al., 2021) are representatives of VR. Since the computer-generated environment should be able to deceive the senses and humans are very sensitive to feedback delays (latencies greater than 50ms are perceptible) (Brooks, 1999), the video must have a high resolution to feel real as well as a high frame rate and very low latency for sensitivity. Consequently, both AR and VR need immense computational power to run properly. However, most mobile devices like smartphones do not have enough computational power to run these applications locally (Yang et al., 2018b). A possible solution is to combine fog computing and mobile AR or VR devices and put intensive computations in the fog. For example, a startup called GridRaster is developing a VR/AR software platform on Saguna's fog computing solution Open-RAN (Alto, 2018) to offer an immersive VR/AR experience on mobile devices by leveraging fog computing. Furthermore, a large shipbuilder, Navantia, is starting to use an AR system, which is also based on fog

[4]http://www.lispmob.org

computing. A study of this system shows that fog computing based AR systems respond clearly faster than cloud-based AR system (Fernández-Caramés et al., 2018).

- **Real-Time Video Analytics:** Real-time video analytics uses Artificial Intelligence (AI) to analyse video contents generated by huge numbers of cameras deployed in buildings, along the streets or in cars in real-time. Fog computing can help many video analytics applications, such as video surveillance, object/face recognition, and smart traffic lights, which typically require low latency, immense computing power and large bandwidth. For example, a bandwidth of 25 Mbps is required to stream a 4K video (Ananthanarayanan et al., 2017). A prototype of a dynamic urban surveillance system based on fog computing, which uses three drones to monitor vehicles and a laptop as the fog node, was built (Chen et al., 2016). Evaluations showed that this scheme is with great promise for smart urban surveillance applications, as it can track the target all the time in a noisy environment, and the estimated speed is close to the actual speed of the target. Furthermore, Ali et al. (2018) show that the efficiency in the throughput of a facial recognition system using deep learning can be considerably improved by putting the initial processing of the data at fog nodes compared to a cloud-only system.

- **Smart Grid:** The smart grid is a virtual network that can control the electricity grid, including energy load, clean energy and grid safety. Many grid control applications are run on the edge of the network, like smart meters and micro-grids (Wei et al., 2014). As grid networks are distributed widely, it is impractical to offload all computational tasks to the cloud. However, the fog can gather and process data from the grid and do real-time analytics and control. For instance, Singh and Yassine (2018) propose and validate an IoT big data analytics system, which uses fog computing to store, process and analyse energy consumption data from smart homes. Furthermore, a fog computing-based application is proposed for power consumption forecasting using smart grid sensors (Jaiswal et al., 2021). In addition, Okay and Ozdemir (2016) proposes a three-tier (i.e., smart meters, fog nodes and cloud data centres) smart grid model and shows that their model improves the cloud computing-based smart grid in terms of privacy, latency, and locality. Finally, fog computing can also be used for distributed demand-side response aggregation in the smart grid (Lyu et al., 2018; Zhu et al., 2019).

- **AVs:** An AV is a vehicle that can drive by itself without human interference, which is the trend in the car industry. For example, all Tesla cars have autopilot features such as lane centring, adaptive cruise control, and self-parking (Ajitha and Nagra, 2021) and Google and Apple are also developing their own AVs. An AV uses cameras, radars and other sensors to capture information from the surrounding environment, which must be processed in real-time to make proper driving decisions. Since AV applications also need very short latency and high bandwidth to send the data, fog computing is particularly suited to run them (Peter, 2015). Here, fog nodes can be AVs or static fog nodes such as smart base stations or smart routers. With fog computing, vehicles without sufficient computing power can utilise these fog nodes to run compute-intensive applications (e.g., AR driving assistance and VR gaming), and spontaneous exchange of information between AVs is made possible (Xiao and Zhu, 2017). For example, a two-level architecture (i.e., AVs and fog nodes) is introduced to cope with the challenges of automated driving services such as large content volume, location-dependence and delay-sensitivity (Yuan et al., 2018). Another framework called vehicular fog computing, which uses moving vehicles such as taxis and buses as mobile fog nodes, is proposed in order to provide on-demand and cost-effective fog computing service for AVs (Xiao and Zhu, 2017). Furthermore, to solve the problem of sensing dead zones, Du et al. (2020) propose a fog computing based architecture for cooperative sensing among adjacent AVs.

1.2 Fog Computing Resource Allocation

A typical fog has computing resources in its fog nodes and networking resources in its network. Computing resources, which generally include processors, random access memory (RAM) and storage, are capable of processing computational tasks. Additionally, IoT tasks are processed by Virtual Machine (VM)s generated in fog nodes. Here, VMs are computer emulations that contain all of the required components to conduct fog tasks. Moreover, there are two different types of networking resources. One is the bandwidth among the fog nodes; the other is the bandwidth between fog nodes and IoT devices. Specifically, fog systems are owned and operated by fog providers, which can be wireless carriers, Internet service providers, cloud service providers and even IoT users who are willing to trade their spare computing resources (Yi et al., 2015). Finally, the main business model of fog computing is the pay-per-use business model, i.e., the fog resources used by IoT users are metered, and they only pay for what they use (Yi et al., 2015).

The RAFC mainly involves three components, which are task scheduling, VM placement and traffic routing (Gu et al., 2018), and DRAFC only involves task scheduling and VM placement. In detail, task scheduling decides when to process each task, and VM placement assigns VMs, which process fog tasks, to appropriate fog nodes. Furthermore, traffic routing finds the paths to send data between IoT devices and fog nodes and between VMs in different fog nodes. Notably, the latter two components are unique to fog computing and do not apply to centralised cloud computing.

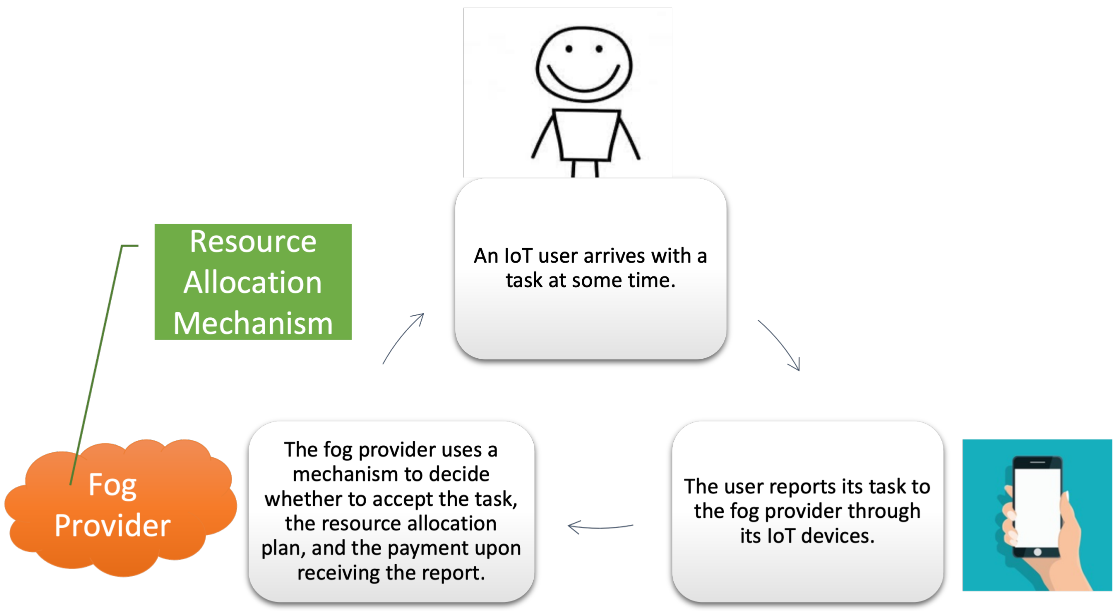
Resource Allocation Mechanism
An IoT user arrives with a task at some time.
Fog Provider
The fog provider uses a mechanism to decide whether to accept the task, the resource allocation plan, and the payment upon receiving the report.
The user reports its task to the fog provider through its IoT devices.

FIGURE 1.3: The procedure of RAFC[5].

Against this background, the procedure of the RAFC and DRAFC is briefly introduced below (Shi et al., 2017) (Figure 1.3). First, the IoT users report the information about their tasks (e.g., value, demanded resource, processing time and deadline) to the fog provider over time. Then, after receiving the report, the fog provider will immediately decide whether to run the task, how much to charge the user, and how to allocate resources for the task. Finally, the fog provider will send these decisions to the user, process the tasks accordingly and wait for new task requirements.

In addition, in a strategic setting, self-interested users can misreport their tasks, such as delaying the report, expanding their processing time, declaring a higher value or an earlier deadline for their tasks. This could cause havoc to the efficiency of a traditional resource allocation mechanism. Therefore, in order to prevent misreporting, we can impose regulations and policies to penalise any users who misreport, or we can take the approach of online mechanism design, which incentivises IoT users to report truthfully. In detail, online mechanism design is a subfield of game theory that studies the problem of how to design mechanisms that decides allocations of resource and monetary transfers based on received reports to get desired objectives when rational users arrive over time (Nisan et al., 2007). We choose to adopt the approach of online mechanism design to deal with this problem because imposing regulations has been criticised as being costly and slow (Demougin and Fluet, 2001). By using an online truthful mechanism, we can be more confident about its efficiency and there is no need to investigate who is violating the regulations.

[5]source of images: `https://www.shutterstock.com/image-vector/hand-holding-smart-phone-vector-illustration-1084761833`

1.3 Research Challenges

After introducing fog computing and the its resource allocation problem, we now discuss the research challenges that we intend to address in this thesis.

1. **Challenge 1** *Limited bandwidth resources.* If bandwidth in the fog is unlimited and free, it is feasible just to consider allocating computational resources for fog tasks. However, in actual situations, bandwidth is limited and costly. Thus, resource allocation mechanisms in the fog need to take bandwidth into consideration and try to make the best of it to reduce cost or latency. By contrast, bandwidth is usually ignored in cloud computing resource allocation (Shi et al., 2016) because the traffic routing is not controlled by the cloud provider (cloud providers just buy bandwidth from Iternet Service Provider (ISP)) (He and Walrand, 2005). However, the emerging software-defined networking (SDN) (i.e., a network approach that enables the network to be centrally controlled using software) (Alamer, 2021) makes centralised network control in fog computing feasible.

2. **Challenge 2** *Dynamic VM allocation.* Another challenge that is unique to fog computing is the dynamic allocation of VMs. First, in our model, VMs are not limited to several types. Instead, they can be dynamically assembled according to users' demands. Second, unlike cloud providers, the fog provider needs to choose a fog node to generate the VM for newly arrived tasks. Furthermore, in order to keep a low latency to a moving IoT device, the VM may need to be moved from one fog node to another every now and then (Bittencourt et al., 2015). Beyond that, VMs in the fog also need to move among fog nodes to serve new tasks efficiently.

3. **Challenge 3** *Time-oriented tasks.* How to allocate time-oriented tasks is another challenge to fog computing. This is because many IoT tasks are time-oriented, meaning they require a specific amount of computational time (between the earliest start time and the task's deadline) to accomplish their maximum value, but they can still achieve a portion of that value if given less time. Consider a user who wishes to utilise a video surveillance application with facial recognition to monitor their stores for 24 hours. If the monitoring lasts less than 24 hours, say 16 hours, it is still useful to them.

4. **Challenge 4** *High social welfare resource allocation.* This challenge is about how to maximise the social welfare (i.e., the overall utility) of IoT users given the limited computing and networking resources of the fog, especially when there are many users demanding diverse resources dynamically. In more detail, the utility of a computing task represents the value obtained by the IoT user from processing

that task. Maximising social welfare is a typical mechanism-design goal because social welfare represents the aggregated satisfaction of users. Thus, the efficiency of resource allocation mechanisms especially means efficiency in terms of social welfare in this thesis.

5. **Challenge 5** *Resource allocation in a strategic setting.* In practice, IoT users may be strategic, which means that users can misreport their tasks for their own benefit. In general, such behaviours are detrimental to the overall efficiency of the IoT users and possibly lead to much worse system performance than expected. To address this problem, the fog provider may use a truthful resource allocation mechanism. Here, truthfulness (or strategyproofness) means that IoT users cannot get a higher utility by misreporting their tasks to the fog provider (Nisan et al., 2007). This challenge is composed of two parts. The first part is how to design a mechanism that achieves truthfulness while only degrading a little proportion of the efficiency or without a loss compared to the best non-truthful mechanism in a non-strategic setting (i.e., users always report their information truthfully unconditionally). The second part is that this truthful mechanism should also outperform state-of-the-art non-truthful mechanisms in social welfare in a strategic setting and outperform state-of-the-art truthful mechanisms in social welfare as well.

6. **Challenge 6** *Resource allocation in online settings.* Furthermore, the online nature of the RAFC, which means that allocation decisions have to be made as information of new tasks reported over time and without any knowledge of the future, makes social welfare maximisation even more challenging. For example, poor myopic decisions can make high-value tasks that arrive in the future unable to be scheduled, which could lead to big losses in social welfare. What is more, it also makes designing truthful mechanisms more challenging. For example, most truthful mechanisms are not truthful anymore in online scenarios. However, a large amount of literature just focuses on offline settings, in which all users report their information simultaneously.

7. **Challenge 7** *Cost-aware resource allocation.* Unlike most existing literature, which solely considers social welfare as the total value of completed tasks, we include resource costs in our model. As a result, our problem, which includes both packing and covering constraints (packing task demands within resource capacities and covering accepted tasks by paying resource operational costs), is more challenging than problems that merely include packing constraints (Azar et al., 2013).

8. **Challenge 8** *Timely resource allocation.* Since it is common to have ad hoc fog tasks that request to process immediately after the report. The resource allocation mechanism should be computationally efficient so that it can make allocation

and pricing decisions right away when it receives new reports from IoT users. However, the resource allocation problem for fog computing has an NP-hard nature (explained in Section 3.1), and the number of fog tasks, fog nodes, and IoT devices can be enormous in the actual world. So it is both important and challenging to design very computationally efficient resource allocation mechanisms.

9. **Challenge 9** *Decentralised resource allocation.* Many fog computing systems may not have a central control to make resource allocation decisions. In this case, a decentralised resource allocation algorithm is needed so that the decentralised fog nodes can make allocation decisions by themselves.

1.4 Research Contributions

In order to address the research challenges listed above, this thesis takes the approaches of constrained optimisation and online mechanism design to study the problem of RAFC with the aim of maximising social welfare. In more detail, constrained optimisation is a domain in mathematical optimisation whose goal is to find the values of related variables to optimise an objective function given some constraints of these variables (Bertsekas, 2014). In our problem, the objective function, which is to be maximised, calculates the total social welfare, and constraints consist of both resource constraints and time constraints. Furthermore, we take the approach of online mechanism design because we intend to design a truthful mechanism, and in our problem, users arrive over time, and thus, resource allocation decisions must be made by an online mechanism.

In addition, this thesis combines reinforcement learning and reverse auction to maximise social welfare for the DRAFC problem, which addresses Challenge 9, 4 and 6. Reinforcement learning is about learning how to take actions to maximise the cumulative rewards (often delayed) (Sutton and Barto, 2018). The trial-and-error search and the delayed reward are the two key aspects of reinforcement learning. In this thesis, a reverse auction is a type of auction in which fog nodes bid for the price at which they are willing to process an analytics task.

The main contribution of this thesis is proposing a truthful online resource allocation mechanism as well as a decentralised mechanism that can achieve near-optimal social welfare for IoT users. By designing, implementing and evaluating our resource allocation mechanism, we show that the truthful mechanism achieves better social welfare than benchmarks and achieves social welfare close to the optimal (around 90%) in a strategic setting. Similarly, the decentralised mechanism can achieve near-optimal social welfare in a decentralised fog system, and its performance is better than benchmarks in terms of social welfare. To address the challenges in Section 1.3, we specifically make the following five contributions.

1. **We formulate the fog computing resource allocation problems (i.e., RAFC and DRAFC) as constraint optimisation problems.** In order to address challenges 1–3 mentioned above, we formulate the fog computing resource allocation problems, which consider the bandwidth constraints and traffic routing as well as allow flexible allocation of VMs, as constraint optimisation problems. For the RAFC problem, we also treat it as an online mechanism design problem in which a fog task requests a certain amount of processing time while meeting certain computing resource demands and time constraints.

2. **We design a truthful mechanism and evaluate its performance in social welfare.** We design a new truthful and individually rational (IR) mechanism called FlexOG. A mechanism is called IR if no participants can get a negative utility by participation (Nisan et al., 2007). Thus, under our mechanism, IoT users will truthfully report their tasks as soon as they get them. We show that FlexOG achieves social welfare better than state-of-the-art benchmarks (up to 10%) and is close to the optimal value (around 90%) using extensive simulations. Therefore, we have addressed challenge 4,6,7 and part one of challenge 5, which is designing a truthful mechanism that outperforms state-of-the-art truthful mechanisms.

3. **We design a scalable and truthful mechanism and evaluate its performance in social welfare.** To improve the scalability of FlexOG, we propose a modified FlexOG called Semi-FlexOG. This mechanism is also truthful and individually rational. Although Semi-FlexOG achieves less social welfare than FlexOG in the simulations (up to 10%), its process time is significantly less than that of FlexOG. This contribution has also addressed challenge 4,6,7 and part one of challenge 5.

4. **We design a decentralised resource allocation algorithm and evaluate its performance in social welfare.** Other than resource allocation in strategic settings, we also consider the decentralised resource allocation for challenge 9. In particular, we propose a decentralised resource allocation algorithm called DAPPO, which combines Multi-Agent Reinforcement Learning (MARL) and online reverse auctions. By extensive simulations, DAPPO is shown to have better performance than the benchmarks in terms of social welfare and can achieve near-optimal social welfare. Similarly, the contribution has addressed challenge 4,6, and 7. Note that the implementation of DAPPO is based on the RLlib[6] library and the *Multi-Agent Deep Reinforcement Learning for Autonomous Vehicles in Disaster Response* project from Jack Parsons[7]. To be specific,

[6]https://docs.ray.io/en/latest/rllib/index.html
[7]https://github.com/jack-parsons

we implement the environement of DRAFC, the benchmark algorithms, the algorithm generating synthetic data and the configurations files for PPO[8].

5. **We conduct simulations to show that our truthful mechanism performs better than the non-truthful online optimal mechanism in a strategic setting.** To fully address challenge 5, we show that FlexOG achieves better social welfare than the online optimal mechanism, which optimally allocates resources over time with the information it has received in a strategic setting by simulations. This is interesting because online optimal achieves better social welfare if all agents report truthfully. We choose online optimal as the benchmark here because online optimal is a greedy algorithm, which is good enough for resource allocation under uncertainty (Gupta et al., 2017). First, we show that, on average, non-truthful agents have a higher utility than truthful agents under the online optimal mechanism, so IoT users indeed have incentives to misreport their tasks. Then, we show that when even a small proportion of agents misreport, the social welfare achieved by online optimal declines significantly and is lower than that achieved by FlexOG[9].

These contributions have led to two peer-reviewed publications:

- Fan Bi, Sebastian Stein, Enrico Gerding, Nick Jennings, and Tom La Porta. A truthful online mechanism for allocating fog computing resources. In *Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, pages 1829–1831, May 2019b. URL `https://eprints.soton.ac.uk/430226/`
- Fan Bi, Sebastian Stein, Enrico Gerding, Nick Jennings, and Thomas La Porta. A truthful online mechanism for resource allocation in fog computing. In *The 16th Pacific Rim International Conference on Artificial Intelligence*, April 2019a. URL `https://eprints.soton.ac.uk/431819/`

1.5 Outline of the Thesis

To let the reader have a clear idea of the structure of this thesis, the following describes its outline. There are five chapters in this thesis. Specifically, in Chapter 2 we outline the literature that is related to our research and the techniques they use to address some of the challenges listed in Section 1.3. It includes the related knowledge of game theory, mechanism design (Section 2.1), reinforcement learning (Section 2.1.3)

[8]`https://docs.ray.io/en/latest/rllib/rllib-algorithms.html#ppo`

[9]We did not use Price of Anarchy (PoA) (i.e., the ratio between the optimal centralised solution and the worst Nash equilibrium) to measure how the efficiency of a system degrades when its agents are self-interested and rational because often pure strategy Nash equilibria do not exist, and randomised strategies are unrealistic in our case.

and MARL (Section 2.1.3.2). Moreover, it also includes the related online resource allocation mechanisms proposed for similar resource allocation problems, such as multiagent resource allocation (MARA) (Section 2.2), Resource Allocation in Cloud Computing (RACC) (Section 2.3) and RAFC (Section 2.4). In Chapter 3 and Chapter 4, we demonstrate the contributions of our research, including the model of RAFC (Section 3.1 and 4.1), the descriptions and properties of benchmarks (Section 3.2 and 4.2) and our proposed mechanisms (Section 3.3.2, 3.3.3, and 4.3), the experimental setup for simulations (Section 3.4.1 and 4.4.1) and results and analysis of the simulations (Section 3.4.2 and 4.4). Finally, we draw conclusions from the research so far and detail our planned future work in Chapter 5.

Chapter 2

Literature Review

Given that the primary objective of our work is to develop a truthful and a decentralised online mechanism that is efficient in terms of social welfare, as described in the previous chapter, this chapter provides the necessary background for our work and relevant literature on both online resource allocation mechanisms for non-strategic and truthful online resource allocation mechanisms. Our main aim is to explore the progress of current research, their advantages and disadvantages and evaluate the extent to which the research challenges in Section 1.3 have already been addressed and what research gaps are still left open.

We begin our literature review with the necessary background knowledge in Section 2.1, including the pertinent concepts of game theory (Section 2.1.1), mechanism design and online mechanism design (Section 2.1.2) because it is the theory of designing mechanisms that achieve a specific outcome, including truthfulness (Challenge 5). Moreover, we present basic knowledge of reinforcement learning in Section 2.1.3 because we use this technique in our decentralised algorithm in Chapter 4. Then in Sections 2.2, 2.3 and 2.4, we examine the work on general MARA, RACC and RAFC, respectively, and discuss what challenges they have addressed and what challenges are still open. Finally, in Section 2.5, we summarise our findings and the main research gaps according to our research challenges.

2.1 Preliminaries

Given that mechanism design is the theory we use to deal with the problem of resource allocation in a strategic setting (Challenge 5), in what follows, we explain the main concepts before proceeding to related work. If the reader is familiar with the basics of game theory and mechanism design, they can choose to skip this section. In particular, mechanism design is a domain in game theory (i.e., the study of agents' strategies and behaviours in a certain game, which means a formal model of an interactive situation). It studies how to design the mechanism, which makes decisions of resource allocation and money transfer, of a game to achieve certain desirable properties or outcomes under the assumption that all agents are rational (Hurwicz, 1973).

Traditionally, game theory and mechanism design are widely used in the field of microeconomics as important analysis tools. Furthermore, they are also used in some computer science problems such as sponsored search auctions (Zhu et al., 2012), spectrum auctions (Qin et al., 2015) and bandwidth allocation (Wu et al., 2012; Xu et al., 2021). Particularly, online mechanism design is an extension of mechanism design that deals with problems in a dynamic environment (Challenge 6) where agents are allowed to come and go at any time, such as the smart grid, cloud computing or fog computing. In contrast, the classic mechanism design only deals with static environments. For example, in a spectrum auction, telecommunications suppliers submit their bids before the deadline, and the spectrum allocation decisions are made all at once. Similarly, in sponsored search auctions, once someone searches online, the advertisers bid for advertisement slots simultaneously. Then a decision is made to decide how all the slots will be assigned.

The structure of this section is as follows. In Section 2.1.1, we introduce the basic concepts of game theory. Then, in Sections 2.1.2 and 2.1.2.3 we examine the relevant concepts and theorems of mechanism design and online mechanism design, respectively.

2.1.1 Game Theory

Game theory is “the study of mathematical models of conflict and cooperation between intelligent rational decision-makers" (Roger, 1991). In more detail, in game theory, decision makers are often called agents, players or participants, and we use the term agents in this thesis. To analyse agents’ behaviours, we must make assumptions about how they behave. In game theory, this assumption is that all agents are rational, which means they always strive to maximise their utility. The utility of an agent is the quantification of its preference over all outcomes of a game, that is, a real number is assigned to each outcome, and this is defined as its utility. If the utility of outcome a is greater than that of outcome b for the agent, then it prefers outcome a to outcome b. In a game, every agent tries to get the best outcome for itself and also knows that other agents similarly try to achieve the best outcomes for themselves. In order to achieve this goal, agents follow strategies that decide their behaviours in all situations. Researchers in game theory have applied these mathematical models to understand decision making in economics (Palafox-Alcantar et al., 2020; Krapohl et al., 2021), biology (McNamara and Leimar, 2020; Dugatkin and Reeve, 2000) and multi-agent systems (Wang et al., 2021; Semsar-Kazerooni and Khorasani, 2009). In particular, RAFC can be modelled as a game, where IoT users are agents of the game, and they act to compete for fog resources (see the details in Section 3.1).

In game theory, a simultaneous game (e.g., rock-paper-scissors) is a game where each agent chooses its actions without any knowledge of the actions of other agents. Our RAFC problem belongs to this type of game because IoT users are assumed not to have any information on other agents' actions, although they do not take actions at the same time. First of all, we present a rigorous mathematical definition of a simultaneous game so that we can introduce other concepts based on it. Formally, we define a simultaneous game as consisting of a finite set I of agents, $I = \{1, 2, \ldots, n\}$. The relevent traits of agent i (e.g., preferences and time constraints) are summarised as the type $\theta_i \in \Theta_i$ of that agent, and the type profile $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ is the set of all agents' types. The behaviour of agent i is captured by a function that maps from all its possible types to its set of actions: $\Theta_i \to X_i$, which is called the strategy of agent i. S_i is the set of all strategies of agent i. The strategy of agent i and the strategies vector of other agents are s_i and s_{-i}, respectively. Then the vector of strategies selected by all agents is $s = (s_1, s_2, \ \ldots, s_n)$. The set of all possible vectors of strategies is denoted as S, and the set of all possible vectors of strategies except agent i's is S_{-i}. The utility of agent i when it plays strategy s_i and others play strategy s_{-i} is $u_i(s_i, s_{-i}) \in \mathbb{R}$.

In addition, the strategies of agents can be categorised into pure strategies and mixed strategies. A pure strategy uniquely determines the behaviour of an agent for any situation (the state of the game) it can face. In other words, an agent using a pure strategy will always take the same action when it faces the same situation. A mixed strategy, on the other hand, is the attribution of a probability to each pure strategy. This enables an agent to choose a pure strategy at random when it is required to take action. In this thesis, we focus on pure strategies because we aim to incentivise agents to always reveal their types truthfully, which is a pure strategy.

Furthermore, given a game, there are different ways to predict its result. We use solution concepts to describe how the game results are predicted, and these predictions are called solutions. There are two fundamental solution concepts: dominant strategy equilibrium and Nash equilibrium. Here, an equilibrium means a strategy profile (i.e., a set of strategies for all agents) that is stable enough to be predicted as the actual result of the game.

To present dominant strategy equilibrium, we first introduce dominant strategies of agents. If an agent has a unique best strategy in the game regardless of other agents' choices, then we call it a dominant strategy for that agent. Namely, $s_i^* \in S_i$ is the dominant strategy for agent i, if:

$$u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i}), \quad \forall s_i \neq s_i^*,\ s_{-i} \in S_{-i}.$$

Furthermore, a weakly dominant strategy for agent i is a strategy such that no other strategies are better than it. Formally, $s_i^* \in S_i$ is a weakly dominant strategy for agent i, if:

$$u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i}), \quad \forall s_i \neq s_i^*, \, s_{-i} \in S_{-i}.$$

Then, a strategy vector $s^* \in S$ is called a dominant strategy equilibrium, if every strategy in it is a weakly dominant strategy:

$$u_i(s_i, s'_{-i}) \geq u_i(s'_i, s'_{-i}), \quad \forall i \in I, \, s'_i \in S_i, \, s'_{-i} \in S_{-i}$$

It is important to realise that when a game has a dominant strategy equilibrium, each agent has a best strategy regardless of the strategies of other players. In other words, each agent's best strategy is unaffected by the actions of other agents. This property is very useful when we design mechanisms because if the dominant strategy equilibrium of a mechanism is desirable, this mechanism can be used in a strategic setting and still get this desirable equilibrium. For example, if we hope that the strategy of every agent is simply to reveal its type truthfully, then the corresponding equilibrium is desirable in this case. Thus, it is the most widely used solution concept in mechanism design, and we focus on designing mechanisms that have dominant strategy equilibria in this thesis.

However, in most games, there is no dominant strategy equilibrium because, in these games, an agent's best strategy changes with other agents' strategies. The stable equilibria in these games are known as Nash equilibria, which is a central concept in game theory with widespread applications. In a Nash equilibrium, no agent can benefit from changing its behaviour unilaterally. Formally, a strategy vector $s \in S$ is called a Nash equilibrium if:

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}), \quad \forall i \in I, \, s'_i \in S_i.$$

Nash equilibria can exist in a stable manner because no agent in the Nash equilibrium has an incentive to change its strategy. So normally, games will end up in one of its Nash equilibria if it has one. This solution concept is not very convincing in predicting agents' strategies if there is more than one Nash equilibrium in a game (Nisan et al., 2007). Furthermore, even when Nash equilibria are unique, they are not certain to be attained because an agent's assumption about the strategies of other agents may be erroneous.

2.1.2 Mechanism Design

We now introduce the core concept of this thesis—mechanism design, which can be viewed as reverse game theory. For game theory, the goal is to analyse the strategies and behaviours of agents in a certain game (as described in Section 2.1.1), while for mechanism design, the goal is to design the mechanism of the game to achieve a certain result or social choice (i.e., an aggregation of different types of agents toward a single collective decision) in a strategic setting (Nisan et al., 2007). Since the types (or part of the types) of the agents are usually private (i.e., not common knowledge), mechanism design is necessary under these circumstances. In particular, the designer of the mechanism does not participate in the game. Instead, it designs the mechanism, or rules of the game, to achieve certain goals (e.g., social welfare maximisation or revenue maximisation). Since the mechanism designer needs to make decisions based on agents' types, the difficulty of mechanism design is how to incentivise agents to reveal their true types.

In order to model strategic behaviours of the agents, a model called independent private values and strict incomplete information game (Nisan et al., 2007, Chapter 9) is adopted in this thesis. Independent private values mean that the valuation of an agent is not affected by any other agents' information. Here, the valuation function of an agent represents its valuation of the result it gets. Unlike the utility function, this does not include any payments (see Definition 2.2). Strict incomplete information means that there is no probabilistic information in the model. For example, we do not know the distribution of valuation functions. This model is reasonable in our RAFC scenario, in which agents are assumed to have independent valuation functions and have no knowledge about the types of others, and the mechanism designer has no probabilistic information about the agents either (Challenge 5). Here, a game comprises a mechanism environment and a mechanism. We first introduce the mechanism environment.

Definition 2.1 (mechanism environment)**.** A mechanism environment $\mathcal{E} = (I, A, \{X_i\}_{i \in I}, \{\Theta_i\}_{i \in I}, \{v_i\}_{i \in I})$ contains the following ingredients:

1. A set of n agents: I
2. A set of results: A
3. For every agent i, a set of actions X_i
4. For every agent i, a set of possible types Θ_i. The type of agent i is $\theta_i \in \Theta_i$
5. For every agent i, a valuation function $v_i : \Theta_i \times X_1 \times \cdots \times X_n \to \mathbb{R}$, which maps from the type of agent i (θ_i) and the actions taken by all agents $(x_1, x_2, \ldots, x_n)$ to a real number (Nisan et al., 2007, Definition 9.40)

In the following sections, we introduce some fundamental concepts and theories in mechanism design. Specifically, we introduce the definitions of social choice function and mechanism (Section 2.1.2.1), and we describe a special type of mechanisms called truthful direct-revelation mechanisms along with an important theory called the revelation principle (Section 2.1.2.2).

2.1.2.1 Social Choices and Mechanisms

Building on the general definitions, we next introduce the notion of a general (nondirect-revelation) mechanism. A general mechanism is composed of two components: an outcome function that chooses a result based on the profile of agents' actions and a payment function that decides a payment for every agent also based on the profile of actions.

Definition 2.2 (mechanism)**.** A general mechanism $\mathcal{M} = (a, \{p_i\}_{i \in I})$ comprises:

1. an outcome function $a : X_1 \times \cdots \times X_n \to A$
2. payment functions $p_1, \ldots, p_n$, where $p_i : X_1 \times \cdots \times X_n \to \mathbb{R}$ (Nisan et al., 2007, Definition 9.24).

Then, the game induced by a mechanism over some mechanism environment is formally defined below.

Definition 2.3 (games of mechanisms)**.** The game $\Gamma(\mathcal{M})$ induced by a mechanism $\mathcal{M} = (a, \{p_i\}_{i \in I})$ over mechanism environment $\mathcal{E} = (I, A, \{X_i\}_{i \in I}, \{\Theta_i\}_{i \in I}, \{v_i\}_{i \in I})$ is the strict incomplete information game $\Gamma(\mathcal{M}) = (I, \{\Theta_i\}_{i \in I}, \{X_i\}_{i \in I}, \{u_i\}_{i \in I})$ where the agents' utility functions $u_i(\theta_i, x_1, \ldots, x_n) = v_i(\theta_i, a(x_1, \ldots, x_n)) - p_i(x_1, \ldots, x_n)$ (Nisan et al., 2007, Definition 9.24).

Usually, we want to make a desirable collective decision based on the preferences of all agents, which can be expressed by a function called the social choice function.

Definition 2.4 (social choice function)**.** The social choice function f maps the types of agents to the results (allocation of goods):

$$f : \Theta_1 \times \cdots \times \Theta_n \to A$$

In general, we are interested in developing a mechanism that produces the same result as a desirable social choice function (e.g., a social choice function that maximises the social welfare based on the preferences of agents). Therefore, if the equilibrium under a mechanism maps to the same outcome as a social choice function, this mechanism implements this social choice function.

Definition 2.5 (implementation)**.** The mechanism implements a social choice function $f : \Theta_1 \times \cdots \times \Theta_n \rightarrow A$ in dominant strategies if for some dominant strategy equilibrium $s_1, \ldots, s_n$ of the induced game, where $s_i : \Theta_i \rightarrow X_i$ we have that

$$f(\theta_1, \ldots, \theta_n) = a(s_1(\theta_1), \ldots, s_n(\theta_n)),\ \forall \theta_1, \ldots, \theta_n \in \Theta$$ (Nisan et al., 2007, Definition 9.24)

In the following section, we introduce a special type of mechanism called direct-revelation mechanisms and a property of mechanisms called truthfulness. Then we introduce the revelation principle, which shows that we can just focus on direct-revelation truthful mechanisms when designing mechanisms.

2.1.2.2 Direct-Revelation Mechanisms and Truthfulness

There is a specific type of mechanism known as direct-revelation mechanisms that is critical to the design of mechanisms due to the revelation principle (see Proposition 2.8). In a direct-revelation mechanism, the set of actions of agent i, X_i, is the set of its possible types Θ_i, which means what an agent can do is just to reveal its type to the mechanism. To formally introduce this definition below, we first introduce some notations. We denote the set of all possible valuation functions for agent i as V_i, and the vector of all types of agents except agent i's as $\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_n)$. Similarly, we define $\Theta_{-i} = \Theta_1 \times \cdots \times \Theta_{i-1} \times \Theta_{i+1} \times \cdots \times \Theta_n$.

Definition 2.6 (direct-revelation mechanisms)**.** A direct-revelation mechanism $(f, p_1,\ \ldots, p_n)$ is made up of a social choice function f and a vector of payment function $(p_1,\ \ldots, p_n)$. The payment functions $(p_1,\ \ldots, p_n)$ maps the reported types of agents to the payments of agents, where

$$p_i : \Theta_1 \times \cdots \times \Theta_n \rightarrow \mathbb{R}$$ (Nisan et al., 2007, Definition 9.14)

Now we introduce the definition of truthfulness, which is a key concept in this thesis. It is also called strategy-proofness or Dominant-Strategy Incentive Compatible (DSIC). Under a truthful mechanism, agent i would prefer providing its true type to the mechanism regardless of other agents' strategies because by providing false information, agent i cannot increase its utility. In other words, truthful mechanisms can prevent agents from misreporting and manipulating.

Definition 2.7 (truthfulness)**.** A mechanism $(f, p_1,\ \ldots, p_n)$ is called truthful or DSIC if $v_i(f(\theta_i, \theta_{-i})) - p_i(\theta_i, \theta_{-i}) \geq v_i(f(\theta_i', \theta_{-i})) - p_i(\theta_i', \theta_{-i}), \quad \forall \theta_i, \theta_i' \in \Theta_i,\ \theta_{-i} \in \Theta_{-i}$ (Nisan et al., 2007, Definition 9.15).

We can now introduce the revelation principle, a crucial principle in mechanism design. It is fundamental because it demonstrates that it is sufficient just to study direct-revelation truthful mechanisms to determine whether social choice function f can be implemented in dominant strategies. This principle makes mechanism design much easier by narrowing the space of mechanisms that need to be searched. For example, if we find the best direct-revelation truthful mechanism in terms of social welfare for a scenario, it is guaranteed to achieve the best social welfare in a strategic setting. Because no general mechanism could achieve better social welfare according to the following revelation principle.

Proposition 2.8. *(Revelation principle) Any social choice functions f that can be implemented in dominant strategies by a general mechanism can also be implemented by a direct-revelation truthful mechanism. Moreover, the payments of the agents in the truthful mechanism are the same as those in the equilibrium of the original mechanism (Nisan et al., 2007, Proposition 9.25).*

In the following, we introduce two desirable properties of a mechanism besides truthfulness in this thesis. The first desirable property is called individual rationality (IR), which guarantees that all (rational) agents are willing to participate. This is required because agents in our RAFC problem cannot be compelled to join the mechanism. To satisfy this property, the mechanism should guarantee that no agents will get a negative utility by joining the game.

Definition 2.9 (IR)**.** A mechanism is individually rational if no agent gets a negative utility by participating the game (i.e., $v_i(f(\theta_1, \ \dots, \theta_n)) - p_i(\theta_1, \ \dots, \theta_n) \geq 0, \quad \forall \theta_1, \dots, \theta_n \in \Theta_n, \ i \in I$) (Nisan et al., 2007, Definition 9.18)

The second desired property is weak budget balance (WBB), which means that the total payment from agents is not lower than the overall cost of the resources. Here, the cost function $o : f(\theta_1, \ \dots, \theta_n) \to \mathbb{R}$ of the mechanism maps from resource allocation decisions to the overall operational cost of the resources (Challenge 7). This property is desired because, normally, fog providers try to avoid losses. So they are not willing to adopt a mechanism that is not WBB.

Definition 2.10. A mechanism is weakly budget balanced if it will never run a deficit. Formally, if for every $\theta_1, \dots, \theta_n$ we have that $\sum_{i \in I} p_i(\theta_1, \ \dots, \theta_n) \geq \sum_{i \in I} o(f(\theta_1, \ \dots, \theta_n))$.

To design truthful mechanisms, we define important properties of truthful mechanisms in the following. These properties can be used to help design mechanisms or judge if a mechanism is truthful.

Proposition 2.11. *A mechanism is truthful if and only if it meets the following requirements $\forall i \in I, \ \theta_{-i} \in \Theta_{-i}$:*

1. *The payment p_i only depends on the outcome of the game $f(\theta_i, \theta_{-i})$, namely, for every θ_{-i}, there exist prices $p_a \in \mathbb{R}$, $\forall a \in A$, such that for all θ_i with $f(\theta_i, \theta_{-i}) = a$ we have that $p_i(\theta_i, \theta_{-i}) = p_a$*

2. *The mechanism optimises for each agent (i.e., $f(\theta_i, \theta_{-i}) \in \arg\max_a (v_i(a) - p_a), \quad \forall \theta_i \in \Theta_i$) (Nisan et al., 2007, Proposition 9.27).*

Then, the following definition describes the property, which is called weak monotonicity (WMON), a social choice function must has in a truthful mechanism. This property means that if the result changes when an agent changes its type, then the agent's value of the new result should be increased relative to its value of the previous result.

Definition 2.12. A social choice function f satisfies WMON if $f(\theta_i, \theta_{-i}) = a \neq b = f(\theta'_i, \theta_{-i}) \implies v'_i(b) - v'_i(a) \geq v_i(b) - v_i(a) \quad \forall i \in I,\ \theta_i, \theta'_i \in \Theta_i,\ \theta_{-i} \in \Theta_{-i}$ (Nisan et al., 2007, Definition 9.28).

Then, the following theorem means that WMON is a necessary condition for truthfulness but becomes a sufficient condition only if all domains of types are convex sets. More specifically, a convex set is a set of points, for every pair of points within it, every point on the straight line that joins them also lies within the set.

Theorem 2.13. *If a mechanism $(f, p_1, \ldots, p_n)$ is truthful, f satisfies WMON. In addition, for every WMON f there exists payment function $p_1, \ldots, p_n$ such that the mechanism $(f, p_1, \ldots, p_n)$ is truthful only if all domains of types Θ_i are convex sets (Nisan et al., 2007, Theorem 9.29).*

2.1.2.3 Online Mechanism Design

In this section, we extend the framework of (offline) mechanism design to online mechanism design, which deals with time-dynamic scenarios where agents' types are reported over time, and decisions must be made without future information. So unlike an offline mechanism, which just makes one overall decision, an online mechanism must make a sequence of decisions over time. Many real-world scenarios are dynamic, such as allocating computational resources to tasks being submitted over time, allocating electrical power to electric cars arriving over time or selling seats on an aeroplane to passengers. The RAFC problem we study in this thesis is also dynamic because IoT users report their tasks over time, and allocation decisions must be made immediately after the reports.

Online mechanism design faces challenges that are different from (offline) mechanism design because the environment is dynamic (Parkes, 2007, Chapter 16). In particular:

1. Decisions must be made without future information.
2. Other than their valuation functions, agents can also misreport their private information about time, such as the arrival times and deadlines of their tasks (Challenge 3)
3. (limited misreports) In practice, an agent may not be able to misreport its type without limitations. For example, in the RAFC scenario, it is impossible to report a task before the agent knows it needs to run this task. So an agent can only misreport a later arrival time of its task.

Next, we discuss the component of a dynamic environment in this thesis. $T = \{1, 2, \ldots, t\}$ denotes discrete (possibly infinite) time steps. Let $q = (q^1, q^2, \ldots)$ denote the sequence of decisions (e.g., accepting/rejecting an agent, the payment of an agent, and when to serve an agent) which can be made when receiving reports from agents. The type of agent i, $\theta_i = (T_i^a, T_i^d, v_i) \in \Theta$, where $T_i^a, T_i^d \in T$ are its arrival time and departure time (deadline), and $v_i(k) \in \mathbb{R}$ is its valuation function. On the other hand, the reported type of agent i is denoted as $\hat{\theta}_i = (\hat{T}_i^a, \hat{T}_i^d, \hat{v}_i) \in \Theta$, which may be different from θ_i because agents can misreport in a strategic setting. The mechanism state $h^t = (\theta^1, \ldots, \theta^t; q^1, \ldots, q^{t-1}) \in H^t$ denotes the mechanism state in time step t, where H^t is the set of possible mechanism states in time step t. It contains the information of agent types $(\theta^1, \ldots, \theta^t)$, and decisions $(q^1, \ldots, q^{t-1})$ that have been made. Furthermore, we use $Q(h^t)$ to denote all possible decisions in state h^t (Parkes, 2007, Section 16.2). In general, the mechahnism state also comprises the stochastic events that occur in the environment, such as new resources are supplied. However, in this thesis, the mechanism state only contains the agent types and decisions because there is no stochastic events in our RAFC problem.

In the following section, we first give a formal definition of limited misreports and the definition of truthfulness under limited misreports, and then we extend the direct-revelation mechanisms and revelation principle to the online domain.

2.1.2.4 Direct-Revelation Online Mechanisms

Now we are ready to introduce the definition of a direct-revelation online mechanism. A direct-revelation online mechanism is an extension of a direct-revelation offline mechanism, but it has more restrictions on agents' behaviours. Note that the mechanism we design for RAFC in Section 3.3.2 belongs to this type of mechanism because the only actions IoT users can take are to report their types, and they report over time instead of reporting simultaneously.

Definition 2.14 (direct-revelation online mechanism)**.** In a direct-revelation online mechanism, each agent only sends a single report about its type to the mechanism.

The mechanism consists a social choice policy $f = \{f^t\}^{t \in T}$ and a payment policy $p = \{p^t\}^{t \in T}$, where decision $f^t(h^t) \in Q(h^t)$ is made in mechanism state h^t and payment $p_i^t(h^t)$ is the money that agent i needs to pay to the mechanism. For convenience, we denote the sequence of decisions as $f(\theta) = (q^1, q^2, \ldots)$ and overall payment collected from agent i as $p_i(\theta) \in \mathbb{R}$, given type profile θ (Parkes, 2007, Definition 16.2).

Unlike offline scenarios, in certain online scenarios, it is reasonable to assume some limitations on the domain of the misreports, which can also simplify the mechanism design problem. For example, agent i cannot report an earlier arrival time than its true arrival time ($\hat{T}_i^a < T_i^a$) or report a later deadline ($\hat{T}_i^d > T_i^d$). This is a very natural assumption because agent i has no idea of its type before its true arrival time T_i^a, and reporting a later deadline will make the task miss its deadline.

Definition 2.15 (limited misreports)**.** The limited misreports that an agent can make is denoted as $C(\theta_i) \subseteq \Theta_i$, where θ_i is the type of agent i (Parkes, 2007, Definition 16.4).

Then, we introduce the definition of a truthful online mechanism given limited misreports, which is similar to the definition of a truthful (offline) mechanism.

Definition 2.16 (truthful)**.** Given limited misreports C, a truthful or DSIC online mechanism satisfies:

$$v_i(\theta_i, f(\theta_i, \theta_{-i})) - p_i(\theta_i, \theta_{-i}) \geq v_i(\theta_i, f(\hat{\theta}_i, \theta_{-i})) - p_i(\hat{\theta}_i, \theta_{-i}),$$

$$\forall \hat{\theta}_i \in C(\theta_i),\ \theta_i \in \Theta_i,\ \theta_{-i} \in \Theta_{-i}$$

(Parkes, 2007, Definition 16.5).

Unfortunately, unlike (offline) mechanism design, only studying direct-revelation truthful online mechanism is not without loss of generality because the online revelation principle does not always hold. However, if the agents are assumed to have no-early arrivals ($\hat{T}_i^a \geq T_i^a$) and no-late departures ($\hat{T}_i^d \leq T_i^d$), then the online revelation principle still holds. Alternatively, the revelation principle can be maintained by requiring each agent to deliver a non-informative "heartbeat" message at each time step $t \in [\hat{T}_i^a, \hat{T}_i^d]$ besides the report of its type (Parkes, 2007, Section 16.2.2). In our RAFC model, although IoT users can report late departures, it is reasonable to request that they deliver a "heartbeat" message at each time step $t \in [\hat{T}_i^a, \hat{T}_i^d]$. So we focus on truthful direct-revelation online mechanisms in our work.

In the following sections, we examine related work on MARA (Section 2.2), RACC (Section 2.3), and RAFC (Section 2.4). Specifically, RAFC is the domain we study in this thesis, RACC is a close domain with many similarities, and MARA is the big domain that RAFC and RACC belong to. We analyse the different resource allocation models they deal with, the techniques they use, and why their approach cannot directly apply to our RAFC problem.

2.1.3 Reinforcement Learning

Reinforcement learning is mainly used to solve problems involving sequential decision-making. A sequential decision-making problem requires the decision maker to make successive decisions based on observations of the environment. Importantly, the objective of reinforcement learning is to maximise the overall return, not the reward associated with a particular decision. Recent advancements in deep reinforcement learning have led to multiple breakthroughs in solving real-world challenges. (e.g., the Game of Go (Silver et al., 2017), StarCraft II (Vinyals et al., 2019) and control of tokamak plasmas (Degrave et al., 2022)).

Next, We introduce basic concepts of reinforcement learning and some commonly used reinforcement learning algorithms in the following sections. Specifically, we introduce basic concepts, different classification methods, and three algorithms (i.e., Q-learning, DQN and PPO) of single-agent reinforcement learning in Section 2.1.3.1. Then, we present basic concepts and different approaches for MARL in Section 2.1.3.2.

2.1.3.1 Single-Agent Reinforcement Learning

In reinforcement learning, we typically use Markov Decision Processes (MDPs) to describe the sequential decision-making processes. Notably, they are referred to as Markov Decision Processe (MDP)s because they satisfy the Markov property (i.e., the next state of a MDP is determined only by the current state and action and is independent of previous states and actions). In reinforcement learning, the Markov property is critical because decisions are made and values are estimated only based on the present state. This is reasonable because agents may not have information of historical states and actions in most scenarios, and many real-life processes satisfy this property.

Formally, we can define MDP as a tuple (S, A, P(.), R(.), h)

- S is a set of states. $s_t \in S$ is the state in step t
- A is a set of actions. $a_t \in A$ is the action in step t
- $P(s' \mid s, a)$ is the state transition function indicates the probability that state s' happens after action a is taken in state s.
- $R(s, a, s')$ is the reward when action a is taken in state s, and the next state is s'
- h is the horizon (a positive integer or infinite)

The MDP runs as follows: The initial state is s_0, and action a_0 is taken. The next state s_1 is based on the transition function $P(s' \mid s, a)$. Then, a_1 is taken and the state transites

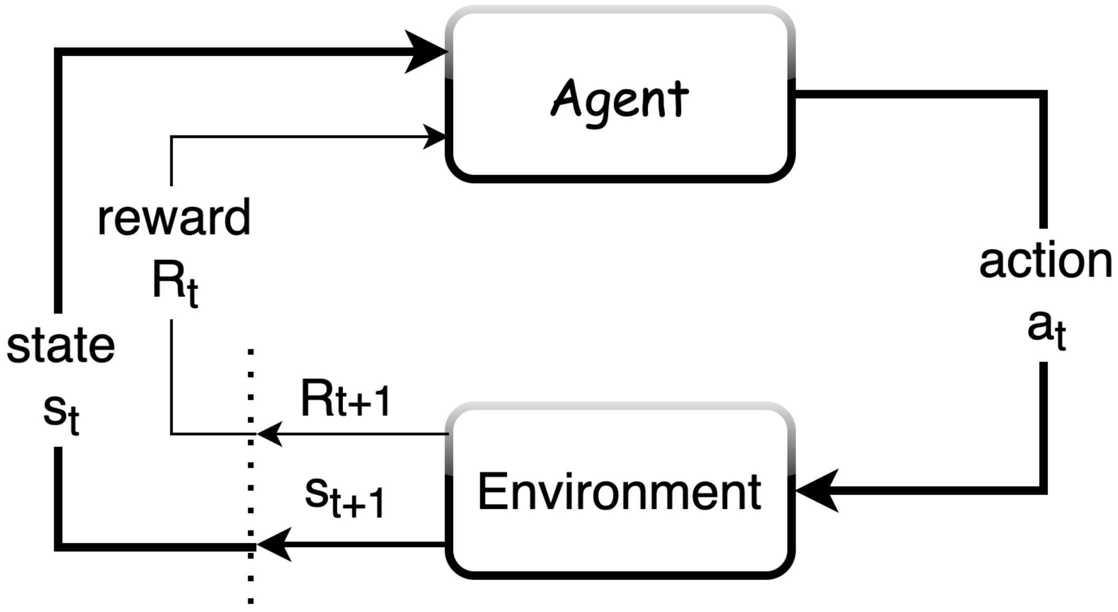

FIGURE 2.1: Markov Decision Process.

to s_2 in the same way. In general, the agent decide action $a_t \in A$ based on state $s_t \in S$ at step t, and the next state s_{t+1} is decided by the trasition function P with a reward $R(s_t, a_t, s_{t+1})$. The loop continues until a terminal state or the horizon h is reached, at which point the episode stops. This process is illustrated in Figure 2.1

It is assumed that the purpose of reinforcement learning is to maximise the overall return. It is defined in Equation 2.1, where $\gamma \in [0, 1)$ is the discount factor (quantifies how much discount we give for future rewards). If γ is big, the immediate rewards are only a little more important than future rewards. However, if γ is a tiny value, the immediate rewards are much more important than future rewards.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

Now, with a definition of MDP, we will introduce what an agent tries to learn in reinforcement learning: a policy. In short, a policy is a rule that an agent uses to decide its actions. Due to the fact that different policies might result in significantly different returns, an agent uses reinforcement learning to find the optimal policy. In formal terms, a policy is defined as the probability distribution across all possible states:

$$\pi(a \mid s) = \mathbb{P}[a_t = a \mid s_t = s]$$

Other than policy π, there are two functions that are especially useful in reinforcement learning. The first function is referred to as the state-value function, and it defines the expected return value of the current state under policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G \mid s_t = s]$$

The second function, called the action-value function or Q-value function, defines the expected return of the current state when action a is taken under policy π:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G \mid s_t = s, a_t = a]$$

Reinforcement learning can be classified into model-based and model-free reinforcement learning, depending on whether the environment's model is to be learned. The agent learns the dynamics of the environment (i.e., the transition function $P(s' \mid s, a)$) through model-based reinforcement learning. Then the agent can get a good policy according to its learned environment model. However, model-based reinforcement learnings are not scalable when the state space or action space grows.

By contrast, a model-free reinforcement learning agent learns through trial and error. Therefore, it does not need to learn the transition function of all possible combinations of states and actions. The algorithms used in this thesis fall into this category.

Additionally, reinforcement learning algorithms can also be classified as on-policy or off-policy depending on how the experiences/samples for learning are generated. For on-policy reinforcement learning, the experiences are generated using the latest learned policy, and then the agent updates its policy using these experiences. For off-policy reinforcement learning, the experiences used for updating policy may not be generated by the latest learned policy. For example, off-policy learning may use all previously collected experiences to update its current policy. Therefore, off-policy learning has better sample efficiency.

Based on what to learn, there are three major families of reinforcement learning algorithms: *value-based*, *policy-based* and *model-based* methods which learn value functions, policies and models, respectively. For example, Q-Learning (Watkins, 1989) is a popular model-free, off-policy, and value-based reinforcement learning algorithm. It is called Q-Learning because it learns the Q-values of all combinations of states and actions, and its policy is to choose the action that maximises the Q-value in any state. Before learning, Q-values could be initialised with arbitrary values. Then, Equation 2.2 is used to update Q-values based on the experiences collected, and it is based on the famous Bellman Equation of Optimality (Equation 2.3)

$$Q^{*}(s_t, a_t) \leftarrow Q * (s_t, a_t) + \alpha \cdot (R_t + \gamma \cdot \max_{a} Q^{*}(s_{t+1}, a) - Q^{*}(s_t, a_t)) \tag{2.2}$$

$$Q^{*}(s, a) = \mathbb{E}_{s'}[R + \gamma \cdot \max_{a'} Q^{*}(s', a')] \tag{2.3}$$

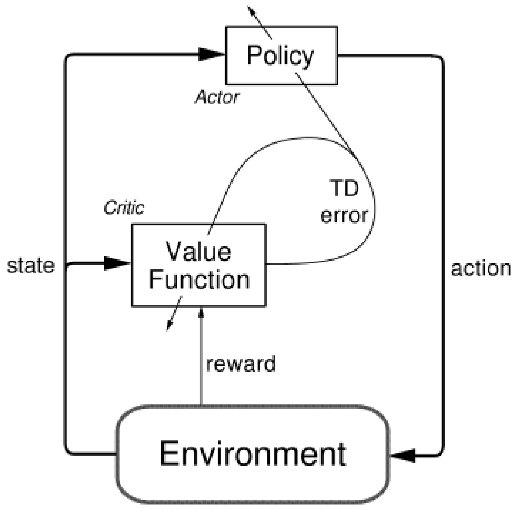

FIGURE 2.2: Actor-Critic architecture (Sutton and Barto, 2018)

where $\alpha \in (0, 1]$ is the learning rate and Q^* approximates the optimal action-value function.

A major limitation of Q-Learning is that it only works in environments where states and actions are discrete and finite. To solve this limitation, Deep Q Network (DQN) (Mnih et al., 2015) introduce Neural Networks to estimate the Q-value function. As a result, a DQN agent is capable of estimating the Q-values of unobserved states. However, with DQN, the action space must be discrete, which is not the case for a large number of problems. To address this issue, we can employ policy gradient approaches, which parameterise the policy function $\pi_t heta$ using a set of parameters $theta$. In this method, we estimate the gradient of the return and use gradient ascent (or descent) to improve the policy parameters θ. Notably, a common architecture of the policy gradient methods is called Actor-Critic (Figure 2.2), where an actor is used to learn a good set of policy function parameters θ and a critic is used to evaluate the policy function estimated by the actor (Sutton and Barto, 2018). Note that both the Critic and Actor are parameterised with neural networks, and Algorithm 1 is the pseudocode for Actor-Critic.

In addition, PPO (Schulman et al., 2017) is a widely used actor-critic reinforcement learning method and is the primary method used in Chapter 4. PPO is said to strike a balance between performance, ease of tunning, and implementation simplicity. The primary goal of this method is to reduce learning variance by ensuring that the updated policy is not far from the previous policy; thus, it is called proximal policy optimization. To introduce PPO, let us first introduce relavent concepts and equations. To begin with, the advantage function $A^{\pi_\theta}(s_t, a_t)$ under a policy π_θ describes how much more reward can be got by taking specific action a_t in state s_t in expectation, compared with taking action according to $\pi_\theta(\cdot \mid s)$, assuming always acting under π_θ afterwards.

Algorithm 1: Actor-Critic Algotithm

```
1  Input: parameters $\boldsymbol{\theta}$ of the policy $\pi_{\boldsymbol{\theta}}(a \mid s)$
2  Input: parameters $\boldsymbol{w}$ of the state-value function $V_{\boldsymbol{w}}(s)$
3  Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0, \alpha^{w} > 0$
4  Initialise parameters $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and $\boldsymbol{w} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
5  for $t = 1 \dots T$ : do
6      $a' \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s)$                                   // sample the next action
7      Take action $a'$ and observe $s', r_t$      // take the action, observe the reward and the next
           state
8      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} V_{\boldsymbol{w}}(s) \nabla \ln \pi_{\boldsymbol{\theta}}(a \mid s)$          // update the policy parameters
9      $\delta_t \leftarrow r_t + \gamma V_{\boldsymbol{w}}(s') - V_{\boldsymbol{w}}(s)$        // compute the Temporal Difference (TD) error
10     $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha^{w} \delta_t \nabla V(s, \boldsymbol{w})$      // update the action-value function parameters
11     $a \leftarrow a'$
12     $s \leftarrow s'$
13 end for
```

Mathematically, the advantage function is defined by

$$A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$$

Then, PPO updates the parameters for policies via:

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{s,a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)]$$

where L is given by

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_{\boldsymbol{\theta}}(a \mid s)}{\pi_{\boldsymbol{\theta}_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a),\ g(\epsilon, A^{\pi_{\theta_k}}(s, a)) \right)$$

where ϵ is a hyperparamter that roughly decides the biggest distance between the new and previous policy and

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A \geq 0 \end{cases}$$

In addition, reward-to-go from t is defined by

$$\hat{R}_t := \sum_{t'=t}^{T} R(s'_t, a'_t, s'_{t+1})$$

Finally, algorithm 2 is the pseudocode of the PPO algorithm.

Algorithm 2: PPO Algotithm

```
1  Input: parameters $\boldsymbol{\theta}$ of the policy $\pi_{\boldsymbol{\theta}}(a \mid s)$
2  Input: parameters $\boldsymbol{w}$ of the state-value function $V_{\boldsymbol{w}}(s)$
3  Initialise parameters $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and $\boldsymbol{w} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
4  for $k = 0, 1, 2, \ldots$ do
5      Collect set of experience $\mathcal{D}_k = \{\tau_i\}$ by using policy $\pi_k = \pi_{\boldsymbol{\theta}_k}$ in the
         environment.
6      Compute rewards-to-go $\hat{R}_t$.
7      Calculate advantage estimations $hatA_t$ based on the current value function $V_{\boldsymbol{w}}$
         (using any method of advantage estimation).
8      Update the policy via stochastic gradient ascent, e.g., Adam (Kingma and Ba,
         2014):
       $$\boldsymbol{\theta}_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left( \frac{\pi_{\boldsymbol{\theta}}(a_t \mid s_t)}{\pi_{\boldsymbol{\theta}_k}(a_t \mid s_t)} A^{\pi_{\boldsymbol{\theta}_k}}(s_t, a_t),\ g(\epsilon, A^{\pi_{\boldsymbol{\theta}_k}}(s_t, a_t)) \right)$$
9      Update the value function via some gradient descent algorithm:
       $$\boldsymbol{w}_{k+1} = \arg\min_{w} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left( V_{\boldsymbol{w}}(s_t) - \hat{R}_t \right)$$
10 end for
```

2.1.3.2 Multi-Agent Reinforcement Learning

Next, we introduce MARL because, in our DRAFC problem (Section 4.1.1), there are multiple fog nodes, and these fog nodes make their decisions independently without central control. Moreover, each fog node can only observe a small part of the full state of the environment. Hence, we introduce Dec-POMDP, which is an extension of Partially Observable Markov Decision Processe (POMDP), and POMDP is a generalisation of MDP. In a POMDP, although the environment dynamics satisfies the Markov property, the agent can only observe part of the state. Hence, the agent has to make decisions under uncertainty of the real environment state, making the problem more challenging. Against this background, Dec-POMDP is a decentralised version of POMDP, which can be formally defined as a tuple $(I, S, A_i, P(.), R(.), \Omega_i, O, h)$ where

- I is the set of agent
- S is a set of states,
- A_i is a set of actions for agent i,
- $P(s' \mid s, a)$ is the state transition function,
- $R(s, a, s')$ is the global reward function,

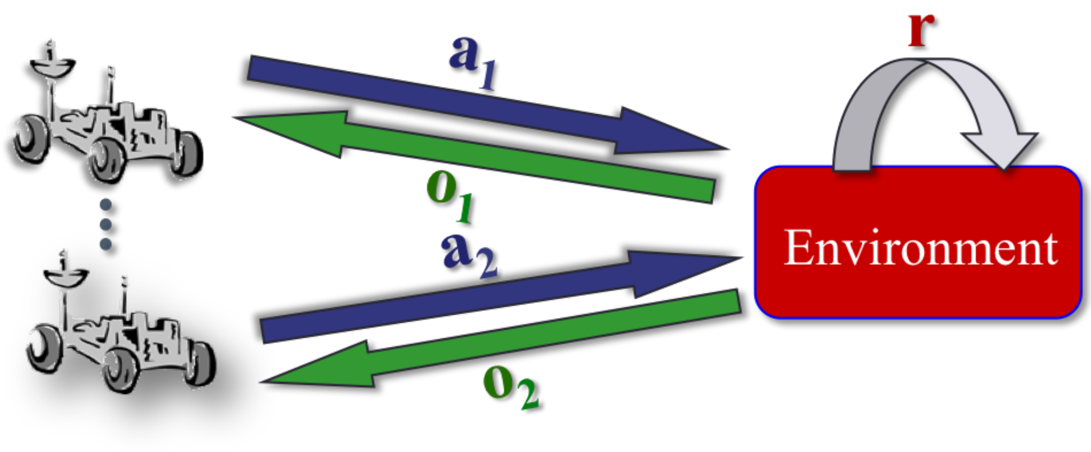

FIGURE 2.3: Model of Dec-POMDP[1].

- Ω_i is the set of observations for agent i
- O is the observation probabilities $O(o \mid s, a) = \mathbb{P}(o \mid s, a)$
- h is the horizon (a positive integer or infinite)

At each time step t of Dec-POMDP, each agent takes an action $A_{i,t} \in A_i$, and the next state is based on the transition function $s_{t+1} = P(s' \mid s_t, a_t)$ (a_t is the joint action at t). Then, each agent gets an observation based on the observation function $O(s_{t+1}, a_t, o)$ and they receive a reward based on the reward function $R(s_t, a_t, s_{t+1})$. These rewards are structured to be fully independent (each agent has its own goal), fully cooperative (all agents have a common goal), or a combination of the two. This process is illustrated in Figure 2.3.

Dec-POMDPs are generally difficult to solve due to the exponential growth of the joint state-action space with the number of agents. Solving such problems, in particular, can require super-exponential time in terms of the number of agents in the worst-case scenario (Bernstein et al., 2002). Hence, we use MARL to learn near-optimal policies for all agents instead of finding the optimal solution directly.

Independent learning is a simple approach of MARL, where each agent uses a single-agent reinforcement learning algorithm (e.g., DQN or PPO), and each agent treats the problem like a MDP without modelling other agents (Tan, 1993). For example, OpenAI Five defeats world champions in an esports game called Dota 2 using PPO (Berner et al., 2019). However, the environment is non-stationary for independent learning because the policies of other agents may change over time. This breaks

[1] `http://rbr.cs.umass.edu/camato/decpomdp/overview.html`

one assumption of MDP that the environment should be static. Hence, independent learning can be unstable and has no guarantee for convergence (Rashid et al., 2018) although it may work fine in some MARL problems in practice (Abed-Alguni et al., 2016).

To solve this problem, an architecture called centralised training and decentralised execution is proposed. In this architecture, we have the full state and actions information during training. Hence we can use this information to train the policies to circumvent the non-stationarity problem. Then, agents use the trained policies in a decentralised way. Two famous algorithms belonging to this architecture are QMIX (Rashid et al., 2018) and MADDPG (Lowe et al., 2017). However, the architecture cannot directly apply to our DRAFC problem because we cannot do centralised training in DRAFC.

2.2 General Multiagent Resource Allocation

In this section, we discuss recent work on general MARA, in which resources are distributed among several agents and these agents may have an impact on the allocation results (Chevaleyre et al., 2006). MARA is relevant to a wide range of applications such as resource allocation in electricity grids (Gradwell and Padget, 2005), network routing (Feldmann et al., 2003), RACC (Wang et al., 2017) and RAFC (Yi et al., 2015), which is the problem we focus on in the thesis. However, the work we discuss in this section does not proactively address many challenges in our RAFC problem (e.g., the dynamic VM allocation challenge (Challenge 2) and the limited bandwidth challenge (Challenge 1)).

Formally, MARA models the following resource allocation problems. There is a set of resources Z and a set of agents I. The resources may be indivisible (e.g., laptops, cars and houses) or divisible (e.g., electricity, RAM and storage). Similar to what is described in Section 2.1.2.2, agents report their preferences over the bundles of resources (i.e., collections of resources that are wrapped together) by means of utility functions (i.e., functions that assign a real number (utility) to each bundle to represent the agents' preferences). The objective is to find the optimal resource allocation plan that maximises social welfare.

Now, other varieties of social welfare exist; some of the most common are utilitarian, egalitarian, and Nash product social welfare. The most common type of social welfare is utilitarian welfare, which is defined as the sum of all agents' utility gains, and it is also the social welfare we try to maximise in this thesis (Challenge 4). This is because we assume that the fog provider is a non-profit organisation and aims to improve the overall utilities.

In terms of utility functions of the MARA problem, some properties (e.g., monotonicity, submodularity, subadditivity, and fractional subadditivity) are to hold because they are realistic for many applications (Nisan, 2000). In the following, we give a formal definition of these properties. Let $u : 2^Z \to \mathbb{R}$ be a utility function.

- u is monotonic if $u(X) \leq u(Y)$ $\forall$ X, Y with $X \subseteq Y \subseteq Z$.
- u is submodular if $\forall$ X, Y with $X \subseteq Y \subseteq Z$ and $\forall x \in Z \backslash Y$ we have that $u(X \cup \{x\}) - u(X) \geq u(Y \cup \{x\}) - u(Y)$.
- u is subadditive if $\forall X, Y \subseteq Z$ we have that $u(X \cup Y) \leq u(X) + u(Y)$.
- u is a fractionally-subaddtive function if there exists a collection of set functions, $\{a_1, a_2, \ldots, a_l\}$ such that each a_j is additive (i.e., the value of $X \subseteq Z$ is the sum of the values of the items in X), and $u(X) = \max_{j=1}^{l} a_j(X)$.

In our problem, the utility function of the agents satisfies three properties of them, except for submodularity (see Section 3.1). This is reasonable for our model because often the processing time near the accomplishment of a fog task is much more valuable than the previous processing time, which causes the utility function not to be submodular.

Furthermore, resource allocation mechanisms can generally be categorised into two main classes, namely, centralised mechanisms and distributed mechanisms. In centralised mechanisms, a central authority is responsible for allocating resources to agents based on their utility functions. A combinatorial auction is a classic example in which bidders compete for bundles of resources, and the auctioneer serves as the central authority. However, centralised mechanisms have the following disadvantages: they require communication between agents and the central authority always to be stable and reliable; the optimal allocation is often computationally hard; the procedure is not very flexible due to its centralised nature (Friedlander, 1982). To address these disadvantages, many fully or partially decentralised allocation mechanisms (Herreiner and Puppe, 2002; Bouveret and Lang, 2011) are proposed. Under a fully decentralised mechanism, agents execute the mechanism totally by themselves, while under a partially decentralised mechanism, agents execute the mechanism with the help of a central authority. However, decentralised allocation mechanisms cannot guarantee the optimality of the allocation results in most situations (Rothe, 2015).

To address the Challenge 5, there is a rich body of work on offline mechanism design where all agents submit their private information and the mechanism only makes a one-time decision (Sandholm, 2003; Hartline and Lucier, 2010; Dobzinski and Dughmi, 2013; Rigas et al., 2020). In particular, a general mechanism which is strategy-proof and maximises social welfare is proposed, called VCG (Vickrey-Clark-Groves) (Nisan et al., 2007). However, offline mechanism design is not suitable for online resource allocation which is essential to our problem.

Hence, online mechanism design, which addresses our Challenge 5 and 6 was introduced by Friedman and Parkes (2003). They extended offline mechanism design to online mechanism design and proposed a strategy-proof online mechanism for allocating WiFi resources. Then, one line of work in online mechanism design proposes online variants of VCG mechanisms (Parkes and Singh, 2003; Parkes et al., 2004; Gershkov and Moldovanu, 2010). In particular, Parkes and Singh (2003) model the online mechanism design problem as a MDP, whose solution can be used to implement optimal policies in a truth-revealing Bayesian-Nash equilibrium. To make the mechanism more scalable, Parkes et al. (2004) adopt sparse-sampling-based MDP algorithm to implement ϵ- efficient policies in a truth-revealing approximate Baysian-Nash equilibrium. Furthermore, Gershkov and Moldovanu (2010) study the truthful welfare maximising allocation of several heterogenous, commonly ranked objects to impatient agents who arrive sequentially. However, these works focus on Bayesian-Nash incentive compatibility and make some restrictive assumptions (e.g., agent types are sampled i.i.d. from a probability distribution known to all agents). In addition, Bayesian-Nash incentive compatibility is weaker because if only all other agents act truthfully then it is also best for an agent to act truthfully. Other types of online mechanisms are also proposed for MARA. For example, Hajiaghayi et al. (2004) design truthful online auctions for allocating identical items. However, they assume the number of users is known in advance. Then, Hajiaghayi et al. (2007) proposed an automated online mechanism design approach when the seller knows the distribution of the bid values. For the problem of allocating identical items with unknown supply, Babaioff et al. (2010) show that no strategy-proof mechanism can achieve better than n-approximation (n is the number of agents) and propose a constant approximation strategy-proof mechanism when the distribution of the supply is known.

Another line of work in online mechanism design considers model-free settings, they have fewer assumptions and propose more tractable mechanisms than VCG variants (Porter, 2004; Hajiaghayi, 2005; Parkes and Duong, 2007). Here, model-free mechanisms are those that do not predict the demand and supply of resources in the future. The advantages of model-free mechanisms are that they need fewer assumptions and are usually easier to compute. Although these mechanisms are myopic (i.e., they make resource allocation decisions without considering possible future situations), their efficiency of resource allocation can still be near-optimal in many scenarios. For example, Hajiaghayi (2005) investigate truthful online auctions for allocating reusable goods for agents who only have unit-length requests and provide two characterisations for strategy-proof mechanisms. Furthermore, a method is introduced to transform a mechanism for online MARA into a truthful one in a discrete-time system and single-valued preference domains (Parkes and Duong, 2007).

The method is called “ironing” because it achieves truthfulness by cancelling allocation decisions that violate monotonicity. However, the main drawback of this method is that it is often computationally difficult because it needs to consider all possible misreports from agents. In addition, Gerding et al. (2011) consider the problem of hybrid Electric Vehicle (EV) charging. Since hybrid vehicles can use both electricity and petrol, they have marginal non-increasing valuations for the electricity charged and have no lower limit of the power charged. To make the mechanism truthful, they combine a greedy algorithm with a technique called “burning", which leaves some units of electricity unallocated when there is still demand unsatisfied. The timing of the “burning" can be at the allocation of the resource or the departure of the agents. In simulations, this mechanism’s performance is close to a non-truthful scheduling algorithm and much better than a fixed price mechanism. However, the “burning" technique may reduce the efficiency of the mechanism significantly. To deal with the hybrid EV charging problem with various charging speeds, Robu et al. (2013) present two greedy-algorithm-based online mechanisms that incentivise agents to truthfully report not only their valuations for the electricity but also their maximum charging rate. The “burning" technique is also used in this literature to achieve truthfulness. Furthermore, a two-sided pricing mechanism, in which both the EVs and charging stations report their private information, is presented (Gerding et al., 2013). However, this mechanism is only truthful on the buyer side (i.e., EVs have no incentive to misreport their types).

Model-based mechanisms, which consider possible future arrivals, are also used by some work to improve the efficiency of the mechanisms (Challenge 4). Stein et al. (2012) study the problem of pure EVs charging and propose a model-based truthful online mechanism to allocate electricity. The mechanism modifies the consensus algorithm (Bent and Van Hentenryck, 2004) by a technique called pre-commitment to achieve truthfulness. In the consensus algorithm, some future scenarios are sampled, and then the scheduling is solved for every one of them. Then, these scenarios vote to decide whether to accept a job (in their case, an EV charging job) or not. A mechanism with pre-commitment will only commit to an agent that the requested resource will be allocated before its departure, but when and how the resource will be allocated remains flexible. Note that this technique is also used in our mechanism for RAFC. Real data simulations showed that this mechanism significantly outperforms a model-free algorithm and is nearly as efficient as an offline optimal algorithm. Moreover, Ströhle et al. (2014) considered the scenario where both the supply and demand for electricity are uncertain. They also developed a truthful online mechanism based on the consensus algorithm, which achieves near-optimal efficiency.

Most of the above literature finds truthful resource allocation mechanisms by looking for an allocation policy that satisfies WMON and then coupling it with a critical-value payment function. However, some literature takes the approach of price-based mechanisms, which decide the resource price before allocation. For instance, Hayakawa et al. (2015) presented a class of price-based truthful online mechanisms that can be used in scenarios where electricity has various marginal generation costs and agents have multi-dimensional preferences by using carefully designed pricing functions and scheduling algorithms. They showed that their mechanism has near-optimal efficiency in a realistic setting with different marginal costs. Moreover, they also present a class of price-based mechanisms allowing increasing marginal valuations, which is DSIC and IR in settings with uncertain procurement costs, multi-unit demand and multi-minded bidders (Challenge 3) (Hayakawa et al., 2018). Although their system model is similar to ours (see Section 3.1) and our work is based on this class of mechanisms, they only assume homogeneous resource. Specifically, they do not consider limited bandwidth resource (Challenge 1) or deal with the dynamical assembly of VMs from several different types of resource (Challenge 2). Therefore, their resource allocation framework has been modified for our RAFC problem in this thesis.

2.3 Resource Allocation in Cloud Computing

In this section, we examine work on RACC, which is a subdomain of MARA and very similar to the RAFC problem we study in this thesis. For example, they all have heterogeneous resources such as CPU, RAM and storage, and agents arrive over time to request VM usage. We mainly focus on the Infrastructure-as-a-Service (IaaS) clouds in this section because the service model of fog computing in this thesis is IaaS (see Section 3.1). Here, IaaS is a service model that serves agents by directly providing computer infrastructures. More specifically, cloud providers pack computational resources (e.g., CPU, RAM and storage) into VMs and provide these VMs to their users. Rather than dynamic VM instances with random VM configurations (Challenge 2), the majority of IaaS cloud providers now offer pre-configured VM instances of fixed types. For example, Amazon Elastic Compute Cloud (EC2) currently offers five categories and 24 types of VMs, and each type has one or more instance sizes[2]. In terms of resource allocation mechanisms, most cloud providers, such as Amazon Web Services (AWS)[3], Microsoft Azure[4], and Google Cloud[5] provide long-term reservation plans or short-range on-demand fixed-price plans to their agents. However, long-term reservation plans are obviously not suitable for agents with uncertain tasks, which is common in IoT scenarios. Although fixed-price mechanisms are easy to implement and are obviously truthful, they are not very efficient in social welfare (Challenge 4) or revenue because they fail to discriminate between different types of tasks (Al-Roomi

[2]https://aws.amazon.com/ec2/instance-types/
[3]https://aws.amazon.com/
[4]https://azure.microsoft.com/en-gb/
[5]https://cloud.google.com/

et al., 2013). For example, a high-value task would not get a high priority in a fixed-price mechanism. Furthermore, they also fail to cater to the volatility of the market, which leads to underpricing or overpricing. To improve efficiency in revenue, spot pricing is also adopted by these cloud providers, such as Amazon EC2[6], Azure Low-Priority VM[7], and Google Preemptible VM Instances[8]. With spot pricing, a cloud agent bids a price for its task. Then the task will run when the spot price is lower than this price, and the agent pays the spot price. However, this mechanism has no guarantee of service-level agreement (SLA). Specifically, the spot price is volatile, and thus, tasks can be interrupted frequently, and there is no guarantee of the finish time of the task. In addition, the spot pricing mechanism is not truthful (Challenge 5) either (Wang et al., 2012b).

To solve the above problems, many mechanisms are designed for RACC in an online manner. For example, Wang et al. (2013); Zhang et al. (2016) propose online auctions for RACC, but they only consider VM instances of a single type and neglect the dynamic allocation of different VMs (Challenge 2). Moreover, a posted pricing mechanism is proposed by Zhang et al. (2017), under which the cloud provider publishes dynamic unit prices for different resource types, and cloud agents either accept the current rates or give up running their jobs. A threshold-based mechanism is also proposed by Farooq and Zhu (2018) to maximise the revenue of the cloud provider. In their model, the cloud provider has a limited number of VMs with different computational efficiency and tasks with different complexity that arrive over time. However, these mechanisms assume that all agents are truthful and do not work in a strategic setting (Challenge 5).

Therefore, a growing body of work is looking at how to efficiently allocate cloud resources to agents in an online and strategic setting, and many truthful mechanisms have been proposed in recent years. The early mechanisms only apply to very simple settings. For example, Wang et al. (2012a); Zhang et al. (2016) propose truthful mechanisms in the setting of single-type VMs, and Lin et al. (2010) treat cloud computing resources as homogeneous. However, this is quite impractical because cloud providers usually have more than one type of VM and more than one type of resource (Challenge 2). Furthermore, Yang et al. (2018a) propose a truthful and envy-free combinatorial auction-based mechanism to maximise the cloud provider's revenue. In their model, there are multiple types of VMs and a single cloud provider and their mechanism combine two ideas: consensus estimate and RevenueExtraction. However, these studies employ Static Resource Provisioning approaches (i.e., the types of VMs are decided before the auction).

To increase efficiency, dynamic VM allocation, where cloud providers can decide how to generate the type and number of VMs, is also studied by some researchers.

[6]`https://aws.amazon.com/ec2/spot/pricing/`
[7]`https://azure.microsoft.com/en-gb/blog/low-priority-scale-sets/`
[8]`https://cloud.google.com/compute/docs/instances/preemptible`

For example, Mashayekhy et al. (2015a) presents a strategy-proof polynomial time approximation scheme (PTAS) mechanism in the scenario of a single cloud provider and they claim their mechanism is the strongest approximation result so far. Furthermore, Nejad et al. (2015) formulate the dynamic VM allocation problem as an Integer Programming problem and design truthful greedy mechanisms to maximise the cloud provider's revenue. Shi et al. (2014a) look at the setting of multi-type VMs and multi-type agents. They use a non-decreasing pricing curve to decide the allocation of the resources and the payments of agents and claim that their truthful mechanism RSMOA is both efficient in the social welfare of the system and the revenue of the cloud provider. Additionally, an online combinatorial auction framework for dynamic RACC computing is proposed by Shi et al. (2014b). This framework is able to model dynamic allocations of heterogeneous VMs and optimise system social welfare over a period of time. Primal-dual optimisation is used to transform a centralised approximation algorithm into a truthful auction mechanism. Additionally, they demonstrated that this auction framework is computationally efficient, truthful, and guarantees a competitive ratio[9]. For the model where an agent requests a specific period of VM, Mashayekhy et al. (2015b) propose a truthful auction-based online mechanism that ensures the users are able to use their VMs for their entire requested period. Considering time-varying user demands, Li et al. (2018) design a truthful online auction-based mechanism, which is composed of a price-based allocation rule and a payment rule, for maximising cloud providers' revenue. Furthermore, Zhang et al. (2018b) propose a truthful heuristic mechanism for a model, where VMs are flexible, and agents can submit multiple requirements, but the cloud provider will at most accept one requirement. However, their research does not examine the temporal correlation in decision-making, as tasks can span multiple time slots (Challenge 3).

Truthful resource allocation mechanisms for tasks with deadlines (e.g., financial companies need data analysis results before the next market open time, or AVs need to identify the object in front of them in a very short time to stay safe) are also studied by some researchers. For instance, Lucier et al. (2013) propose two deadline-aware algorithms for homogeneous resource (i.e., CPU time) allocation in two settings. In one setting, the tasks can be paused and resumed (preemptible), but in the other setting, once a task is started, it must be processed until completion. They design a mechanism using the dual fitting technique for the first setting and prove its competitive ratio as well as showing that it significantly outperforms other heuristics used in practice. For the other setting, they prove that no performance guarantee can be given and propose an efficient heuristic called COMMITTED for resource allocation. Finally, they prove that their COMMITTED algorithm can be made truthful very easily. Additionally, preemptive scheduling with the assumption of deadline slackness (i.e., the lower bound on the ratio of a task's required time to the time window in which it can be

[9]The competitive ratio is defined in this context as the worst ratio between the social welfare cost incurred by an online mechanism and the optimal social welfare.

processed) has also been studied (Azar et al., 2015). A truthful online mechanism with a guaranteed competitive ratio for the overall utility of jobs is presented in their work, and another truthful online mechanism is developed that can commit whether a job will complete before its deadline if the deadline slackness is large enough. In similar work (Zhou et al., 2017), cloud agents not only specify their deadlines for the job but also provide penalty functions for the situation that the deadlines are violated. An auction framework with posted prices is used to incentivise truthful reports, and to address the soft deadline limitations, a novel technique of compact exponential-size LPs paired with dual separation oracles is proposed. Moreover, the classic primal-dual framework is used to develop a social welfare approximation algorithm, and their mechanism is shown to be efficient by simulations using Google cluster data (Reiss et al., 2011). In addition, Zhang et al. (2020) consider time-varying multidimensional resource allocation in clouds and design a truthful online auction to achieve high social welfare, short execution time and high resource utilisation. Finally, Babaioff et al. (2022) focus on machine learning jobs, whose utilities decline with completion time, and design strategy-proof and constant competitive mechanisms to allocate in a cloud.

However, all the above work considers the setting of cloud computing, in which a single centralised data centre provides resources, so their techniques do not deal with bandwidth resources (Challenge 1) or dynamic VM allocation (Challenge 2). Furthermore, they do not consider the operational cost of cloud providers, which is also different from our setting (Challenge 7).

2.4 Resource Allocation in Fog Computing

Although fog computing is a relatively new concept, many studies have been carried out on resource allocation in this domain. As mentioned in Section 1.1, the fog computing structure is three-tier, which consists of the IoT devices, the fog, and the cloud. Some work on resource allocation only considers the fog (Singh and Singh, 2021; Aazam and Huh, 2015), while others consider the fog and IoT devices (Zeng et al., 2016) or the fog and cloud simultaneously (Cao et al., 2021; Agarwal et al., 2016). However, we only look at resource allocation in the fog tier in this thesis.

Notably, there are many performance metrics in RAFC and different studies may try to optimise different performance metrics. The following list shows some performance metrics that are commonly used.

- **Execution Time:** The execution time is the time required to complete the tasks, excluding any waiting time.
- **Latency:** Latency is the time taken to acquire the processed results of a task from the fog. It is the summation of computational time and transmission time.

- **Makespan:** Makespan is the overall time required to complete a workflow (i.e., a series of dependent tasks).
- **Deadline:** Deadline is about the percentage of tasks that finish before their deadlines.
- **Throughput:** Throughput is the number of tasks processed per unit time.
- **Energy Consumption:** Energy consumption is the amount of energy used to process all tasks.
- **Social Welfare:** Social welfare is the overall utilities got by processing the tasks and is the performance metric of this thesis.

Next, we will have a review of the studies in RAFC, and we categorise them by their methods. In detail, the main methods are traditional methods, integer linear programming, heuristics and reinforcement learning. Traditional methods are static algorithms (e.g., First-Come-First-Served (FCFS), Min-Mim and Min-Mam) that can be used when all the information is known before resource allocation. For example, Bittencourt et al. (2017) utilise three traditional scheduling methods, namely FCFS, concurrent, and delay-priority, to optimise the scheduling of mobility-aware applications. In another study, Mtshali et al. (2019) analyse four algorithms (i.e., Round-Robin, FCFS, SJF, and Genetic Programming) in terms of energy consumption, average task delay, network utilisation, and execution time and conclude that FCFS performs the best in fog computing task scheduling.

An integer linear programming problem is a constraint problem in which the constraints are linear, and some or all of the decision variables are restricted to be integers. When some of the decision variables are not discrete, the problem is referred to as mixed-integer linear programming (Schrijver, 1998). Furthermore, Hoseiny et al. (2021) formulate the Fog resource allocation problem as a mixed-integer nonlinear programming problem and propose two scheduling algorithms called *Min-CCV* and *Min-V*. In detail, *Min-CCV* is for minimising computation, communication and violation cost, and *Min-V* is for minimising deadline violation cost. Their extensive simulations show their proposed algorithms can significantly improve deadline satisfaction and decrease the total cost. To minimise the overall service request latency in hybrid Fog-Cloud computing, Aburukba et al. (2020) formulate the IoT service scheduling problem as an integer linear programming problem and propose a customised genetic algorithm. Their algorithm is shown to have lower overall latency compared with three traditional methods.

Heuristic is a method that can find a good enough solution to a problem quickly, although the solution may not be optimal. A heuristic-based fog computing job scheduling algorithm is proposed to lower the latency and network usage (Jamil et al., 2020). By iFogSim simulations, their algorithm has better performance in latency and network usage compared with FCFS method. In addition, Auluck et al. (2019) propose two heuristic algorithms for scheduling tasks with stringent deadlines. In general, more latency-sensitive tasks are processed on fog nodes, and less latency-sensitive tasks are processed on the cloud. Simulation results show their approach increases the success ratio and decreases the average response time compared to scheduling tasks on the cloud alone. Furthermore, Aazam and Huh (2015) introduces an efficient resource allocation framework that predicts and reserves resources based on agents' historical behaviour and offers prices based on agents' characteristics. Their simulations show that their framework can allocate resources adaptively and avoid resource wastage. For joint resource allocation in fog and cloud computing,

Meta-heuristic (e.g., evolutionary algorithms, simulated annealing, ant colony optimisation and bee life algorithm) is a problem-independent algorithm to find heuristic optimisation algorithms (Hussain et al., 2019). For instance, an improved firework algorithm is proposed to decrease the execution time of fog tasks and to enhance the load balance of fog nodes (Wang et al., 2020). They enhance the firework algorithm by incorporating an explosion radius detection mechanism and demonstrate that their technique outperforms the benchmarks. Hyper-heuristic is a method that automatically selects, combines, generates or adapts several simple heuristics to solve optimisation problems (Burke et al., 2013). Kabirzadeh et al. (2017) introduce a hyper-heuristic scheduling algorithm for fog computing. Simulations using iFogSim show that their algorithm has lower energy consumption and cost than that of benchmarks (i.e., simulated annealing, particle swarm optimisation algorithm and ant colony optimisation algorithm). Additionally, the hybrid-heuristic algorithm combines two or more heuristic algorithms to obtain superior performance to any single heuristic (Baraglia et al., 2001), for example, Wang and Li (2019) propose a hybrid-heuristic-based task scheduling algorithm that combines improved particle swarm and enhanced ant colony algorithm in order to reduce energy consumption and delay while improving the efficiency of smart production lines using fog computing.

Moreover, the resource allocation problem becomes more challenging considering the inherent dynamism and uncertainty of the fog computing system (e.g., random arrival of different types of analytic tasks and dynamic changes). Hence, the conventional optimisation methods may not allocate resource in fog very efficiently. To further increase the efficiency of resource allocation, reinforcement learning is also proposed by many studies because of its ability to self-learn and handle the ever-changing environment. To reduce power consumption and job execution latency in fog computing, an ϵ-greedy Q-learning task offloading algorithm is proposed by Liu et al. (2019). Specifically, each IoT device learns a policy to decide whether to offload its tasks to the fog nodes or not. They demonstrate through extensive simulations that their strategy provides an improved trade-off between task execution latency and power consumption. Furthermore, deep Q-network is also adopted to find the optimum policy for resource allocation in fog (Yu et al., 2017; Chen et al., 2018; Zhang et al., 2018a). Additionally, a scheduling algorithm based on double deep Q-learning is proposed with the aim of lowering computation cost, service delay, and energy consumption for fog-based IoT applications (Gazori et al., 2020). Their evaluation shows their proposed algorithm outperforms the benchmarks. Tuli et al. (2020) schedule applications in an edge-cloud computing system using another reinforcement learning method termed asynchronous-advantage-actor-critic (A3C) with residual recurrent neural networks. They claim their approach can quickly adapt to dynamic environments, and has significantly better performance in energy consumption, running cost and response time than state-of-the-art algorithms. However, many approaches focus on centralised mechanisms, where a centralised fog controller learns the optimal allocation policy for analytic tasks. To make the resource allocation more robust and scalable, allocation mechanisms using multi-agent reinforcement learning (MARL) are also proposed (Zhang et al., 2019b; Liu et al., 2020). In these mechanisms, each end user learns its policy to offload its tasks to the fog independently. However, not all IoT devices have the computational ability to do reinforcement learning in practice.

In particular, the above work does not consider the strategic behaviour of agents (Challenge 5). Thus, in a strategic setting where agents act according to their utilities, these mechanisms can no longer guarantee their efficiency. Although there is little work that directly studies the truthful online mechanism for RAFC, some literature has looked at related domains such as distributed cloud computing, Cloudlets and edge computing, which have similar resource allocation models. First of all, we discuss the work on resource allocation in distributed clouds. Similar to RAFC, resource allocation mechanisms in distributed clouds need to decide not only when but also where to put the VMs (Challenge 2) and the bandwidth between VMs (Challenge 1). Shi et al. (2015) have studied the problem of bandwidth allocation in a distributed cloud. They developed truthful offline and online mechanisms from Shapley value based auctions. Although real data simulations show that their mechanisms are efficient, their mechanism is only for bandwidth allocation. Moreover, Zhang et al. (2015) design truthful online auctions where agents bid for VMs for a fixed time in the future for social welfare and revenue maximisation. This research utilised an online primal-dual optimisation approach to maximise social welfare and a randomised reduction algorithm to transform the social welfare maximisation auction to a revenue maximisation auction. They showed that their mechanisms are polynomial-time (Challenge 8) and have better performance than existing ones using Google cluster data (Reiss et al., 2011). However, they did not consider the bandwidth between servers and between server and agents (Challenge 1), and their agents are not multi-minded (Challenge 3). Furthermore, two truthful online mechanisms are proposed by Shi et al. (2017) for dynamic virtual cluster (VC) provisioning. VCs are assemblies of several VMs and the communication resources (bandwidths) between them. They designed the Social Welfare Online Mechanism (SWMOA) to maximise social welfare as well as the Provider Revenue Maximization Online Auction (PRMOA) to maximise cloud providers' revenue. SWMOA's core idea is to maintain a dynamic virtual unit cost for each resource type and accept a requested VC scheme (i.e., the placement of VMs) only if its total virtual cost is less than its valuation. PRMOA, on the other hand, first determines a provisional VC allocation and payments for agents using SWMOA and then charges each accepted bid with a randomised boosted payment, which is still below its valuation, to increase the cloud providers' revenue. However, under their mechanisms, cloud agents need to specify their desired schemes of VC placement, which is not practical because cloud agents usually have no knowledge about the following things: the physical topology of the distributed cloud and the desirable VC schemes. Additionally, Zhang et al. (2019a) consider offloading tasks from IoT devices to mobile devices, and propose a truthful online rewards-optimal auction, which is based on Lyapunov optimisation and VCG auction, to optimise social welfare. However, their mechanism is an offline auction and thus does not allow agents to dynamically join the auction in real time. For the scenario of vehicular fog computing, Peng et al. (2020) propose a truthful multiattribute-based (e.g., location, reputation, and computing power) double auction mechanism for reasonable matching. For the scenario of edge computing, Gao et al. (2019) study the problem of allocating VMs in geo-distributed Edge Cloud Nodes and propose a truthful auction-based mechanism, which consists of a greedy winning bid selection algorithm and a critical payment pricing algorithm. Finally, Aggarwal et al. (2021) consider the Fog-Integrated Cloud Architecture and propose a truthful reverse auction where the fog provider and cloud provider are participants. Their mechanism is multi-attribute, polynomial time, and achieves a low resource procurement cost. However, the above three mechanisms are offline too.

2.5 Summary

In this chapter, we have presented the concepts and theories in game theory, mechanism design, and reinforcement learning, as well as the literature that is related to our work. As discussed beforehand, the research community has proposed many effective approaches for RAFC and other related resource allocation problems. However, they only address a subset of our challenges.

First, we examined related knowledge in game theory (Section 2.1.1), mechanism design (Section 2.1.2) and reinforcement learning (Section 2.1.3) as a background. In particular, we introduced important concepts such as game, solution concept, mechanism, and we gave the definition of a special type of mechanism called direct-revelation truthful mechanisms. Then we showed that we could just focus on this type of mechanisms because of the revelation principle. Furthermore, we listed desirable properties for our mechanism in this thesis, such as truthfulness, IR and WBB. Finally, we discussed fundamental concepts of reinforcement learning and presented some popular reinforcement learning algorithms.

Second, we presented background and related work in general MARA in Section 2.2. In the background part, we introduced the general model of MARA and different types of utility functions, objective functions and resource allocation mechanisms and clarified which type our work focuses on. Then we examined online mechanisms proposed for MARA. Although they address parts of our challenges in isolation, they typically fail to consider heterogeneous resources (Challenge 1 and Challenge 2).

Finally, we examined the related work in RACC or RAFC, which also addresses a subset of our challenges. The main problem of work in RACC is that it usually does not deal with bandwidth resources (Challenge 1) or location-aware VM allocation (Challenge 2). Some work in RAFC considers most of our challenges, but they still have significant shortcomings, such as requiring agents to define every detail of VM location. As a result, no effort has been made to develop a truthful online mechanism capable of resolving all of our challenges in RAFC. Thus, this is one of the research gaps that we intend to bridge in this thesis. The other research gap is that most of the studies using reinforcement learning propose centralised resource allocation algorithms and fail to address Challenge 9. In particular, some insights from the examined work are integrated into our work. We adapt the price-based mechanisms (Hayakawa et al., 2018) and the pre-commitment technique (Stein et al., 2012) to our RAFC problem and design our mechanism belonging to this class of mechanisms because price-based mechanisms are guaranteed to be DSIC and IR.

To evaluate our work, we choose some mechanisms as benchmarks: the offline optimal mechanism represents the upper limit of social welfare efficiency; the online optimal indicates the social welfare could be achieved if all agents report truthfully; the online greedy algorithm is an online truthful mechanism based on the idea from (Gerding et al., 2011); SWMOA2 is a variant of a state-of-the-art truthful mechanism called SWMOA (Shi et al., 2017). Random allocation and bidding zero are two decentralised algorithms that can be used in the decentralised resource allocation. The details of these benchmarks are described in Section 3.2 and 4.2.

Chapter 3

Truthful Resource Allocation Mechanisms in Fog Computing

In this chapter, we present our work on designing a truthful online mechanism for resource allocation in fog computing (RAFC), which addresses our Challenges 1-7 in Section 1.3. In Section 3.1, we describe the model of our RAFC problem. Then, in Section 3.3.1, we introduce a class of truthful online resource allocation mechanisms call price-based mechanisms to which our proposed mechanism belongs. Next, we present the algorithms of benchmark mechanisms in Section 3.2 and the algorithm and properties of our mechanism in Section 3.3.2. After that, we show the setup and the results of the simulations in Section 3.4. Finally, Section 3.5 summarises the whole chapter.

3.1 Model of RAFC

In this section, we present the RAFC model used in this thesis. We first give a brief overview of this model in Section 3.1.1, and then formally describe it in detail in Section 3.1.2.

3.1.1 Overview of the Model

In our model, a single fog provider owns a fog computing system with several geographically distributed fog nodes and data links interconnecting them, as shown in Figure 3.1. Here, fog nodes have computational resources (such as CPU, RAM and storage) and data links have bandwidth resources, and different resources have different fixed operational costs. The operational costs of computational resources comprise electricity costs and the depreciation charge of the resources, and the

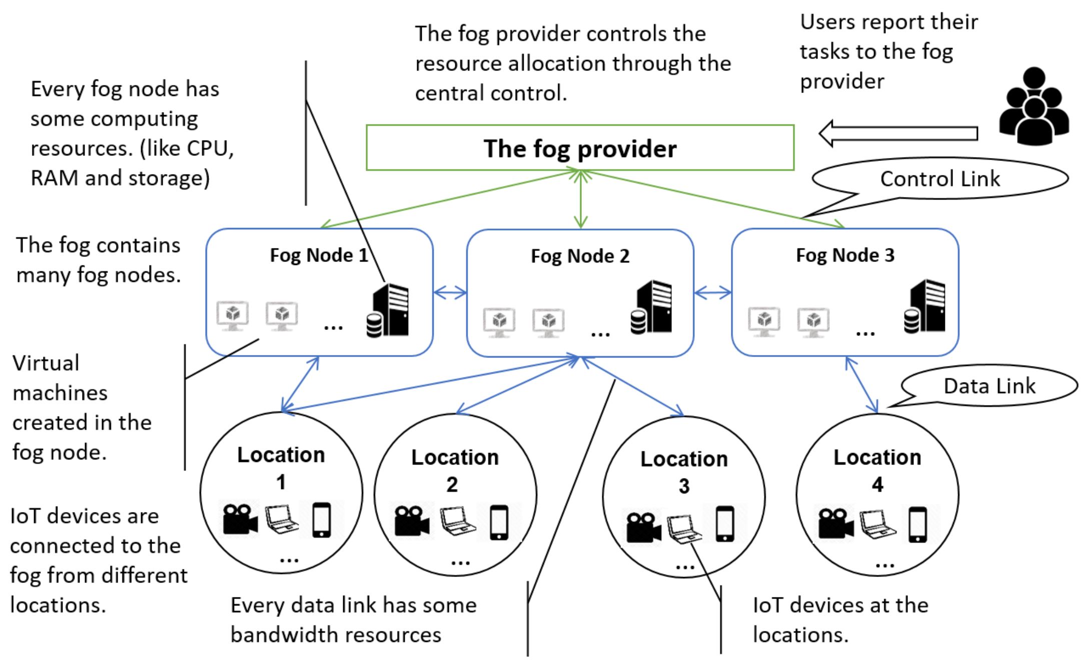
The fog provider controls the resource allocation through the central control.
Users report their tasks to the fog provider
Every fog node has some computing resources. (like CPU, RAM and storage)
The fog provider
Control Link
The fog contains many fog nodes.
Fog Node 1
Fog Node 2
Fog Node 3
Virtual machines created in the fog node.
Data Link
Location 1
Location 2
Location 3
Location 4
IoT devices are connected to the fog from different locations.
Every data link has some bandwidth resources
IoT devices at the locations.

FIGURE 3.1: General view of a fog computing system.

operational costs of bandwidth are the costs charged by the ISPs. Fog nodes and data links jointly supply these resources to fulfil the demands of IoT users by way of processing tasks using VMs. IoT devices at different locations are connected to fog nodes of the fog through data links. They can be stationary IoT devices in the fog computing system or portable IoT devices carried by IoT users. For example, stationary IoT devices can be smart TVs, surveillance cameras and smart speakers, while IoT devices carried by agents can be smartphones and AVs. Further, the fog provider manages the resource allocation of the fog via a central point of control and control linkages. More specifically, the central point of control is a server, which receives analytics tasks requests from IoT devices/agents and decides how to allocate resources to satisfy these agents and the payment for each agent, and these decisions are sent to fog nodes to execute through control links.

Another essential part of our model is IoT users. They report their tasks (e.g., video surveillance tasks or picture processing tasks) to the fog provider through their IoT devices over time, which includes the resource requirements, the time constraints and the valuation functions of the tasks. In particular, we assume that their tasks are pre-emptive and may migrate to other fog nodes during execution. Furthermore, some users can always get the value of their tasks if the results are given to them before the deadline, while others must get real-time results. For example, a picture processing application can have computing tasks like adding filters or compressing photos. To finish the task of adding filters to 100 photos, bandwidth resources are needed to send the images to and from the fog node, and computational resources are needed to process the image. The time constraints of this task could be like this: the task can start right now, and the result should be sent to the agent in 30 seconds because agents usually do not want to wait for too long to add a filter to their photos. If the task is completely finished before the deadline, the agent can get all the value of this task. However, if filters have only been added to 50 photos at the deadline, the agent can still get part of the value.

When receiving the reports from the applications, the fog provider decides whether to accept them, how to allocate resources (i.e., computation time at a certain configuration) to satisfy the demands of the accepted tasks and how much the corresponding payment is through an online mechanism. Finally, the social welfare of the allocation is the sum of value IoT users get by processing tasks minus the sum of the fog provider's operational costs, while the revenue is the sum of the fog provider's income minus its operational costs. We focus on social welfare because we assume that the fog provider is a non-profit organisation, and improving overall social welfare is its primary objective. Therefore, we leave the objective of maximising the fog provider's revenue to future work.

3.1.2 Formal Model of RAFC

We now present the model described in Section 3.1.1 in detail and formally. Consider a fog provider with a set W of geo-distributed fog nodes and a set L of locations, which are interconnected through a set $\mathbb{E}$ of data links, as shown in Figure 3.1.

Furthermore, there is a set E_l of IoT devices in each location l, and $e \in E_l$ is an IoT device (e.g., a smart TV, surveillance camera, smart speaker or AV). Every fog node $w \in W$ has a set R of limited computational resources (e.g., CPU, RAM and storage). Moreover, there are $A_{w,r}$ units of type $r \in R$ resources in fog node w, and the unit operational cost of resource r in fog node w is $o_{w,r}$. In addition, the bandwidth capacity and the unit operational cost of link $(j,k) \in \mathbb{E}$ are $b_{j,k}$ and $o_{j,k}$ respectively. For simplicity, we assume that the bandwidth capacity and unit bandwidth costs are symmetrical for all links (i.e., $b_{j,k} = b_{k,j}$, $o_{j,k} = o_{k,j}$, $\forall (j,k) \in \mathbb{E}$). Fog nodes and data links together offer their resources to satisfy the needs of IoT users. In particular, we assume that VMs can be created in a fog node to run computing tasks as long as there are enough computational resources in that fog node, and the total resource requirements of several VMs are just the sum of their individual resource requirement for simplicity, although, in reality, they may need fewer resources because they can share resource with each other. Furthermore, the fog provider uses a centralised online resource allocation mechanism to make resource allocation and payment decisions.

IoT users with tasks arrive over time, and I denotes the set of all tasks. Note that we adopt a continuous time system, but the tasks can only start execution at discrete time steps, denoted by the set $T = \{1, 2, \dots, |T|\}$. Each task $i \in I$ is owned by an agent, which is also denoted as i for simplicity because we assume that each agent has one task. In addition, the arrival time of task i is $T_i^a \in [0, |T|]$, which is the time when agent i becomes aware of its task i, and the time window that the task can be processed is from its start time T_i^s to its deadline T_i^d. Here, we assume that no tasks arrive at the exact same time, and agent i reports its task's type $\hat{\theta}_i$ (as defined in the following) at time $\hat{T}_i^a$ to run a certain application (e.g., a video surveillance application or a picture processing application). We assume that agent i wants to know the number of time steps $\tilde{t}_i$ it will get and the payment p_i for its task also by time $\hat{T}_i^a$ because agents want to run the tasks locally or elsewhere if their tasks get rejected. The operational cost of task i is denoted as o_i, which is the sum of costs of all resources allocated to task i, including the cost of bandwidth. Furthermore, we also assume that every task only requires one VM to run but may require connections to several endpoints $e \in E$ (in the same location or in different locations) because this is common in an IoT system (Du et al., 2018). IoT users are also assumed to be stationary, which means that the IoT devices of agents do not change locations over time. Furthermore, we also assume VMs can migrate between fog nodes and the migration costs are negligible, and all tasks are pre-emptive (i.e., all tasks can be interrupted and restarted), which means that they can always be paused and resumed. Finally, we focus on one type of task called time-oriented tasks (e.g., video surveillance and video processing tasks), which are common in fog computing. Such a task i needs a certain configuration of resources for a time length t_i to get its full value but can still get part of the value if the processing time is less than t_i. Formally, the type of task i is a tuple $\theta_i = (T_i^a, T_i^s, T_i^d, v_i, \{a_{i,r}\}_{r \in R}, \{\Gamma_l^i\}_{l \in L})$, where $a_{i,r}$ denotes the amount of resource $r \in R$ required, and Γ_l^i denotes the bandwidth demand between its VM and location $l \in L$. For simplicity, bandwidth demands are symmetrical. That is, Γ_l^i denotes both the bandwidth demands to and from location $l \in L$. In this paper, the valuation function is given by $v_i = \{v_{i,0}, v_{i,1}, \dots, v_{i,t_i}\}$, where $v_{i,t}$ is the value when task i gets a usage time of t time steps within its time window ($[T_i^s, T_i^d]$) and ti denotes the amount of time required to complete the task. We reasonably assume that the valuation function increases monotonically with usage time(i.e., $v_{i,t''} \geq v_{i,t'}$, $\forall t'' \geq t'$). For instance, assume an agent wants to use real-time video analytics combined with facial recognition to monitor their stores 24 hours a day. It is intuitive that they will not get additional value for a surveillance time of more than 24 hours, and it is nevertheless beneficial to them if the surveillance is fewer than 24 hours in duration, for example, 18 hours. We pick this sort of valuation function because it is applicable to a large number of IoT applications that gain more utility as processing time grows. Additionally, the reported task type i is a tuple $\hat{\theta}_i = (\hat{T}_i^a, \hat{T}_i^s, \hat{T}_i^d, \hat{v}_i, \{\hat{a}_{i,r}\}_{r \in R}, \{\hat{\Gamma}_l^i\}_{l \in L})$, and $\hat{\theta}^{\langle t \rangle}$ represents the set of all types reported up to and including the time step t.

Now, because a key assumption of our study is that agents are strategic, $\hat{\theta}i$ may not be identical to θ_i. Additionally, we assume limited misreports (see Section 2.1.2.4) due to the nature of our problem (i.e., $\hat{T}_i^a \geq T_i^a, \hat{T}_i^s \geq T_i^s, \hat{T}_i^d \leq T_i^d, \hat{a}_{i,r} \geq a_{i,r}\, r \in R, \hat{\Gamma}_l^i \geq$

$\Gamma_l^i\, l \in L$). This is reasonable because an agent cannot report a task before it becomes aware of it (i.e., $\hat{T}_i^a < T_i^a$), and cannot report a looser time window (i.e., $\hat{T}_i^s < T_i^s$ or $\hat{T}_i^d > T_i^d$) because the fog provider can idendty whether task i is ready at $\hat{T}_i^s$ and send the processed results to i at $\hat{T}_i^d$. Thus, reporting $\hat{T}i^s < Ti^s$ will be identified and will result in the task being cancelled, and reporting $\hat{T}i^d > Ti^d$ will result in zero value. Finally, agent i would not misreport a lower resource demand because its task cannot be processed in that situation. However, for some tasks, the results cannot be withheld until their deadlines, such as video surveillance tasks and autopilot tasks. This situation is discussed in Section 3.3.4.

Following that, when the fog provider receives the bid $\hat{\theta}i$ for task i, it will determine the resource allocation plan λ_i, which is a tuple formally defined in the later part of this paragraph, right away to this task. The decision comprises the amount of usage time $\tilde{t}_i$ to be allocated, as well as the payment p_i because of the assumption we made earlier that all agents want to know the allocation results at their arrival times. To give an upper bound on social welfare, we solve a constraint optimisation problem in an offline scenario to determine the optimal social welfare, with the decision variables: (1) $\{z_{w,t}^i \in \{0,1\}\}_{i\in I, w\in W, t\in T}$, indicating whether task i's VM is placed in fog node w ($z_{w,t}^i = 1$) or not ($z_{w,t}^i = 0$) in time slot t. (2) $\{f_{l,w,j,k,t}^i \in \mathbb{R}_{\geq 0}\}_{i\in I, l\in L, w\in W, (j,k)\in \mathbb{E}, t\in T}$, indicating the bandwidth allocation on link (j,k) for traffic from location i to fog node w associated with task i at time step t. Thus, given task i, its duration of execution $\tilde{t}_i = \sum\limits_{w\in W, t\in T} z_{w,t}^i$, and its resource allocation plan $\lambda_i = (\{z_{w,t}^i\}_{i\in I, w\in W, t\in T}, \{f_{l,w,j,k,t}^i\}_{i\in I, l\in L, w\in W, (j,k)\in\mathbb{E}, t\in T})$. Finally, the objective function (3.1a) of this optimisation problem maximises the social welfare, which is subject to resource and time constraints:

$$\max_{\lambda_i} \quad \sum_{i\in I} v_i(\sum_{w\in W, t\in T} z_{w,t}^i) - (\sum_{i\in I, r\in R, w\in W, t\in T} a_{i,r} z_{w,t}^i o_{w,r} + \sum_{i\in I, l\in L, w\in W, (j,k)\in\mathbb{E}, t\in T} 2 o_{j,k} f_{l,w,j,k,t}^i) \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{w\in W} z_{w,t}^i \leq 1,\ \forall i \in I, t \in T \tag{3.1b}$$

$$\sum_{i\in I} z_{w,t}^i a_{i,r} \leq A_{w,r},\ \forall w \in W, r \in R, t \in T \tag{3.1c}$$

$$z_{w,t}^i = 0,\ \forall i \in I, w \in W, t < T_i^s\, or\, t > T_i^d \tag{3.1d}$$

$$\sum_{j:(j,w)\in\mathbb{E}} f_{l,w,j,p,t}^i = \Gamma_l^i z_{w,t}^i,\ \forall w \in W, i \in I, l \in L, t \in T \tag{3.1e}$$

$$\sum_{k:(l,k)\in\mathbb{E}} f_{l,w,l,k,t}^i = \Gamma_l^i z_{w,t}^i,\ \forall w \in W, i \in I, l \in L, t \in T \tag{3.1f}$$

$$\sum_{j:(j,k)\in\mathbb{E}} f_{l,w,j,k,t}^i = \sum_{j:(k,j)\in\mathbb{E}} f_{l,w,k,j,t}^i,\ \forall w \in W, k \in W, i \in I, l \in L, t \in T \tag{3.1g}$$

$$\sum_{i\in I} f_{l,w,j,k,t}^i \leq b_{j,k},\ \forall (j,k) \in \mathbb{E}, t \in T \tag{3.1h}$$

$$f_{l,w,j,k,t}^i \geq 0,\ \forall i \in I, l \in L, w \in W, (j,k) \in \mathbb{E}, t \in T \tag{3.1i}$$

To explain the above constraints in detail, constraint (3.1b) represents that every task only needs one VM. Constraint (3.1c) guarantees that the allocated resources at any fog node will not surpass its resource capabilities at any time slot. Constraint (3.1d) means that the VM is created within the start time and deadline of the task. Constraint (3.1e) requires that the total inbound traffic to fog node w for the traffic from task i's location l equals its corresponding bandwidth demand at each time step if the VM for task i is placed in fog node w. Since the bandwidth demands and the bandwidth costs are both symmetrical. (i.e., The task requires the same upstream bandwidth and downstream bandwidth and the unit operational cost for upstream and downstream bandwidth is also the same.) It is sufficient to just consider the traffic from L to W. Constraint (3.1f) indicates that the outbound traffic from task i's location l is equal to its corresponding bandwidth demand at each time step. Constraint (3.1g) represents that the inbound and outgoing traffic of intermediary nodes should be equal for task i. Constraint (3.1h) ensures that aggregated traffic on each traffic link does not exceed its bandwidth capability at any point in time. Finally, constraint (3.1i) ensures that the allocated bandwidth in each data link is not negative, which is impossible in practice.

This is a mixed-integer linear programming problem because the constraints and the objective function are linear and one of the decision variables $z^i_{w,t}$ is discrete. Unfortunately, this problem is NP-hard, which can be proved by reducing a 0-1 knapsack problem to it (see detailed proof below). In practice, the optimisation problem (3.1a) can be solved using linear programming solvers. In particular, we utilise the IBM ILOG CPLEX Optimization Studio in this thesis. However, solving this problem can be time-consuming because the problem is hard per se.

Theorem 3.1. *The optimisation problem (3.1a) is NP-hard.*

Proof. Suppose we have a 0-1 knapsack problem with a maximum weight capacity A_w and $|I|$ items with weights $\{a_i\}_{i\in I}$ and values $\{v_i\}_{i\in I}$. We can design a fog computing resource allocation problem as follows. There is one fog node with CPU resource A_w and no other resources. All $|I|$ tasks arrives at $t = 0$ and task i requires only a_i CPU resource with start time $T^s_i = 0$, deadline $T^d_i = 1$, and value $v_i, i \in I$ for $t_i = 1$. Then, we can solve the above 0-1 knapsack problem by just solving the above fog computing resource allocation problem. Ergo, the offline optimal problem is at least as hard as the 0-1 knapsack problem. Since the optimisation problem of a 0-1 knapsack problem is NP-hard, the optimisation problem (3.1a) is NP-hard too. □

Finally, mechanisms need to make both resource allocation decisions and payment decisions for IoT users. We use $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle}) \in \mathbb{R}_{\geq 0}$ to denote the task i's payment, which is a function of the resource allocation (λ_i) and all types of tasks received by $\hat{T}^a_i$ $(\hat{\theta}^{\langle \hat{T}^a_i \rangle})$. Thus agent i's utility is $u_i = v_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle})$, and this is what agent i tries to maximise.

3.2 Resource Allocation Benchmarks

Against this background, we present the benchmark algorithms and the mechanism we proposed for RAFC. We describe these algorithms in detail and show the pseudocode of them except for offline optimal because for the offline optimal algorithm, what we need to do is just solving the optimisation problem of all tasks.

3.2.1 Offline Optimal Algorithm

We assume in this algorithm that we have complete information about future tasks and allocate resources to maximise social welfare without the need to incentivise IoT users to report their tasks truthfully. The social welfare of this theoretical and idealised scenario is computed via solving the constraint optimisation problem 3.1a in Section 3.1.2.

3.2.2 Online Optimal Algorithm

This algorithm is akin to the offline optimal algorithm, except that at each time step, the optimisation problem is performed using only the tasks that have arrived thus far (and not of future tasks). While this algorithm is not entirely truthful, we use it to estimate the amount of social welfare that could be produced in an online situation if all agents reported truthfully. In Section 3.4.2.4, we also evaluate this algorithm's performance in social welfare in settings where part of the agents misreport. The details of this algorithm are given in Algorithm 3 below.

3.2.3 Online Greedy Algorithm

This algorithm myopicly makes an allocation decision to maximise the social welfare of a task with the earliest time slots possible when it arrives. Then it commits to the decision henceforth. Moreover, it calculates the payout based on the task's associated operational costs ($p_i = o_i = \sum_{r \in R, w \in W, t \in T}(a_{i,r} z^i_{w,t} o_{w,r}) + \sum_{l \in L, w \in W, (j,k) \in \mathbb{E}, t \in T}(2 o_{j,k} f^i_{l,w,j,k,t})$). This mechanism is DSIC and IR, and the details of this mechanism are given in Algorithm 4:

Theorem 3.2. *The OG mechanism is DSIC, IR and WBB.*

Proof. Under the OG mechanism, any agent who gets allocated nothing has to pay zero because the corresponding operational cost is zero. So it satisfies the third condition of Definition 3.5. Furthermore, for every possible $\tilde{t}_i$ allocated to task i, the mechanism chooses the allocation that has the lowest o_i according to $\hat{\theta}_i$. This is because $\hat{v}_i(\tilde{t}_i)$ is

Algorithm 3: The online optimal algorithm

```
1  $\Theta_{arrived} \leftarrow \emptyset$                                  // The set of arrived tasks
2  $\theta_{flex} \leftarrow \emptyset$                                     // The set of flexible tasks
3  $\Lambda \leftarrow \{\ \}$                  // The dictionary of allocation decisions for each task
4  for $t$ in $T$ do
5      while new tasks arrive within $t$ do
6          when a new task $i$ arrives                                    // Tasks arrive over time
7          $\Theta_{arrived} \leftarrow \Theta_{arrived} \cup \{i\}$              // Update the set of arrived tasks
8          $\theta_{flex} \leftarrow \theta_{flex} \cup \{i\}$                    // Update the set of flexible tasks
9      end while
10     solve the maximum utility allocation for tasks in $\theta_{flex}$ (i.e.,
        $\arg\max_{\lambda_j} \sum_{j \in \theta_{flex}} (\hat{v}_j(\lambda_j) - o_j(\lambda_j)))$ // Find the allocation for tasks in $\theta_{flex}$ that maximise
        their social welfare
11     $p_i \leftarrow 0$                                                  // Payment for task $i$ is zero
12     for $i$ in $\theta_{flex}$ do
13         if the next time step $(t+1)$ is allocated to i according to $\lambda_i$ then
14             $\Lambda[i] \leftarrow \Lambda[i] \cup \lambda_{i,(t+1)}$   // Save the allocation decision of the next time step for
                task $i$
15             $t_i \leftarrow t_i - 1$                          // Update the remaining usage time of task $i$
16             for $j$ = 1; $j \leq t_i$; $j$++ do
17                 $\hat{v}_i(j) \leftarrow \hat{v}_i(j+1)$              // Update the valuation function of task $i$
18             end for
19         end if
20         if $t_i = 0$ then
21             $\theta_{flex} \leftarrow \theta_{flex} \setminus \{i\}$   // Delete task $i$ from $\theta_{flex}$ if it gets its required usage time
22         end if
23         if $t = \hat{T}_i^d$ then
24             $\theta_{flex} \leftarrow \theta_{flex} \setminus \{i\}$         // Delete task $i$ from $\theta_{flex}$ if it reaches its deadline
25         end if
26     end for
27 end for
```

independent of the allocation details, and the mechanism maximises $\hat{v}_i(\tilde{t}_i) - o_i$. Since $p_i = o_i$, the payment p_i is also independent of $\hat{v}_i$. In addition, increasing $\hat{T}_i^a, \hat{T}_i^s$ or decreasing $\hat{T}_i^d$ can only increase o_i by reducing the space of possible allocations (i.e., reducing the available time steps (increasing $\hat{T}_i^s$ or decreasing $\hat{T}_i^d$) or causing more resource to be allocated to other agents (increasing $\hat{T}_i^a$)), and increasing $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$ or $\{\hat{\Gamma}_l^i\}_{l \in L}$ can only increase o_i too because this increases the resource demands. Due to the fact that $p_i = o_i$, $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ is monotonic according to Definition 3.4. Hence, the mechanism satisfies the condition 1 in Definition 3.5. Finally, the mechanism also satisfies the condition 2 in Definition 3.5 because it decides the allocation λ_i that maximise $\hat{v}_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ so the $\tilde{t}_i$ incurred by λ_i also maximises the utility of agent i. Taken together, the OG mechanism is DSIC and IR according to Theorem 3.6. In addition, because the payment of task i equals to the operational cost of that task ($p_i = o_i$), the total sum of payments equal the total operational cost of the fog

Algorithm 4: The online greedy mechanism

```
1  $\Theta_{arrived} \leftarrow \emptyset$                      // The set of arrived tasks
2  $\Lambda \leftarrow \{ \}$                 // The dictionary of allocation decisions for each task
3  for $t$ in $T$ do
4      while new tasks arrive within $t$ do
5          when a new task $i$ arrives                     // Tasks arrive over time
6          $\Theta_{arrived} \leftarrow \Theta_{arrived} \cup \{i\}$       // Update the set of arrived tasks
7          solve the optimal utility allocation for task $i$ (i.e.,
             $\arg\max_{\lambda_i} \sum_{t \in T} (\hat{v}_i(\lambda_i) - o_i(\lambda_i))$, s.t. constraints in Problem 3.1a and given $\Lambda$ &
             $\hat{\theta}_i$)      // Find the allocation for task i that maximise its social welfare
8          choose the $\lambda_i$ whose time slots sum is the smallest if the previous problem
             has multiple solutions.      // Choose the allocation with the earliest time slots
9          $\Lambda[i] \leftarrow \lambda_i$                          // Commit this allocation decision
10         $p_i \leftarrow o_i(\lambda_i)$         // The payment for task i is its corresponding operational cost
11     end while
12     allocate resources in time step $(t+1)$ according to $\Lambda$
13 end for
```

$(\sum_{i \in I} p_i = o)$. Thus, OG satisfies WBB. □

3.2.4 SWMOA2 Algorithm

While the SWMOA mechanism described by Shi et al. (2017) cannot be directly applied to our model, we build a variation named SWMOA2 as a viable benchmark. The major distinction between this mechanism (described in Algorithm 5) and OG is that it maintains a virtual cost for each resource rather than an operational cost. For convenience, we will refer to M as the set of all computational resources available at each fog node and bandwidth resources available on each link, with m denoting one type of them. To determine the virtual costs, we define the load factor $\kappa_{m,t}$ as the percentage of occupied resource m at time step t. The virtual cost is then: $c_{m,t} = \mu^{\kappa_{m,t}} - 1$, $\forall t \in T, m \in M$, where $\mu = 2|M|F + 2$ and F is the upper limit of the ratio of the highest and lowest unit time task valuations.

Then, task i's virtual cost is $c_i = \sum_{r \in R, w \in W, t \in T} (a_{i,r} z^i_{w,t} c_{w,r,t}) + \sum_{l \in L, w \in W, (j,k) \in \mathbb{E}, t \in T} (2c_{j,k,t} f^i_{l,w,j,k,t})$. Thus, the agent can use resources cheaply when resources are abundant and is restrained when resources are in shortage. In the original paper by Shi et al. (2017), the virtual cost also prevents allocations that violate the resource constraints, which no longer works in our model, because, unlike them, we do not assume an upper bound of each task's resource requirements. Therefore, resource constraints are added to this mechanism. SWMOA2 is also DSIC and IR and the detail of it is shown in Algorithm 5 below.

Theorem 3.3. *The SWMOA2 mechanism is DSIC and IR.*

Algorithm 5: The SWMOA2 mechanism

```
1  $\Theta_{arrived} \leftarrow \emptyset$                                   // The set of arrived tasks
2  $\Lambda \leftarrow \{\ \}$                                  // The dictionary of allocation decisions for each task
3  $\kappa_{m,t} \leftarrow 0, \forall m, t$                    // The load factors of resources
4  $c_{m,t} \leftarrow 0, \forall m, t$                        // The virtual costs of resources
5  for $t$ in $T$ do
6      while new tasks arrive within $t$ do
7          when a new task $i$ arrives                       // Tasks arrive over time
8          $\Theta_{arrived} \leftarrow \Theta_{arrived} \cup \{i\}$   // Update the set of arrived tasks
9          solve the maximum virtual utility allocation for task $i$ (i.e.,
             $\arg\max_{\lambda_i}(\hat{v}_i(\lambda_i) - c_i(\lambda_i))$, given $\Lambda$ & $\hat{\theta}_i$)   // Find the allocation that maximises
             task $i$'s virtual social welfare
10         $\Lambda \leftarrow \lambda_i$                      // Commit this allocation
11         $p_i \leftarrow c_i(\lambda_i)$                    // The payment for task $i$ is its virtual cost
12         $\kappa_{m,t} \leftarrow \kappa_{m,t} + z^i_{w,t} a_{i,r} / A_{w,r}, \forall m \in P \times R, t \in T$   // Update the load factors of
             computational resources
13         $\kappa_{m,t} \leftarrow \kappa_{m,t} + \sum_{l \in L, w \in W} f^i_{l,w,j,k,t} / b_{j,k}, \forall m \in \mathbb{E}, t \in T$   // Update the load factors of
             bandwidth resources
14         $c_{m,t} = \mu^{\kappa_{m,t}} - 1, \forall t \in T, m \in M$   // Update the unit virtual costs of resources
15     end while
16     allocate resources for the next time step $(t+1)$ according to $\Lambda$
17 end for
```

Proof. Following a similar argument, we can prove that SWMOA2 satisfies conditions one and three in Definition 3.5. Furthermore, the payment p_i is also independent of $\hat{v}_i$ because the virtual cost c_i does not depend on $\hat{v}_i$. In addition, $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle})$ is also monotonic because increasing $\hat{T}^a_i$ not only results in resource being allocated to other agents but also increases the virtual costs of resources, increasing $\hat{T}^s_i$ or decreasing $\hat{T}^d_i$ still reduces the available time steps to allocate, and increasing $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$, $\{\hat{\Gamma}^i_l\}_{l \in L}$ increases the resource demands. Therefore, the SWMOA2 mechanism satisfies all the conditions in Definition 3.5 and is also DSIC and IR by Theorem 3.6.

□

3.3 Proposed Resource Allocation Mechanisms

We begin this section by characterising a class of truthful resource allocation mechanisms in Section 3.3.1, and then list all benchmarks in Section 3.2. At last, we describe our truthful resource allocation mechanisms in Section 3.3.2 and 3.3.3.

3.3.1 Price-based Mechanisms

To begin, we present price-based mechanisms that ensure DSIC and IR for our resource allocation problem, and it includes our proposed mechanism. Note that this class of mechanisms is an adaptation from the price-based mechanisms characterised by Hayakawa et al. (2018). In the following, we characterise the properties that the price-based mechanisms should have in order to guarantee DSIC and IR.

Definition 3.4. A monotonic payment function is one that increases (weakly) monotonically over $\hat{T}_i^a$, $\hat{T}_i^s$, $\hat{t}_i$, $\hat{a}_{i,r}, r \in R$ and $\hat{\Gamma}_l^i, l \in L$, and decreases (weakly) monotonically over $\hat{T}_i^d$.

Then, we define the class of price-based mechanisms for RAFC as follows:

Definition 3.5 (monotonic price-based mechanisms)**.** If an online mechanism meets the following criteria, it is classified as a monotonic price-based mechanism:

1. The mechanism computes the payment pi for any potential resource allocation plan λ_i to task i using a payment function $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ that is monotonic and independent of $\hat{v}i$.
2. Task i's resource allocation plan λ_i maximises $\hat{v}_i(\lambda_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ (over all possible λ_i for task i with any $\hat{v}_i$).
3. Payment is zero for tasks that do not get allocated.

Then, the following theorem assures that any price-based mechanisms are DSIC and IR.

Theorem 3.6. *For our RAFC problem, any online mechanism that satisfies Definition 3.5 is DSIC and IR.*

Proof. The proof is shown below. □

To prove a mechanism that satisfies Definition 3.5 is DSIC, we first use the sufficient and necessary characterisation of incentive compatible mechanisms for a setting where agents can only misreport their valuation function (i.e., where the misreports of $\hat{T}_i^a$, $\hat{T}_i^s$, $\hat{T}_i^d$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$ and $\{\hat{\Gamma}_l^i\}_{l \in L}$ are not considered) proposed by Bartal et al. (2003) .

Lemma 3.7. *In our setting, assuming the parameters $\hat{T}_i^a$, $\hat{T}_i^s$, $\hat{T}_i^d$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$, $\{\hat{\Gamma}_l^i\}_{l \in L}$ in agent i's bid $\hat{\theta}_i$ are truthful, a direct revelation mechanism is DSIC if and only if*

1. *The payment function $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ is computed for every possible allocation $\tilde{t}_i(\lambda_i)$ for task i and does not depend on $\hat{v}_i$.*

2. *The allocation function allocates $\tilde{t}_i$ for task i such that the value of $\hat{v}_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ is maximised (over all $\tilde{t}_i$ that can be allocated to i for any choice of $\hat{v}_i$).*

Lemma 3.7 is a simple extension of Theorem 1 in (Bartal et al., 2003).

Then, we use this Lemma to prove the following theorem that is in settings where agents can misreport $\hat{T}_i^a, \hat{T}_i^s, \hat{T}_i^d$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$, $\{\hat{\Gamma}_l^i\}_{l \in L}$ besides $\hat{v}_i$ (assuming limited misreports).

Theorem 3.8. *In our setting, a direct revelation mechanism is DSIC if and only if*

1. *The payment function $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ for every possible allocation $\tilde{t}_i(\lambda_i)$to task i is monotonic and does not depend on $\hat{v}_i$.*

2. *The allocation function allocates $\tilde{t}_i$ for task i such that the value of $\hat{v}_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ is maximised (over all $\tilde{t}_i$ that can be allocated to i for any choice of $\hat{v}_i$).*

Proof. At first, we show that the conditions in Theorem 3.8 are sufficient. According to Lemma 3.7, an agent i cannot increase its utility by manipulating $\hat{v}_i$. Therefore, we can assume that it truthfully reports its valuation coefficient $\hat{v}_i$, and the only way to increase its utility is by decreasing the payment function. Since the payment function is monotonic, only misreporting $\hat{T}_i^a < T_i^a, \hat{T}_i^s < T_i^s, \hat{T}_i^d > T_i^d$, $\hat{t}_i < t_i$, $\hat{a}_{i,r} < a_{i,r}, r \in R$ or $\hat{\Gamma}_l^i < \Gamma_l^i, l \in L$ can reduce it. First of all, misreporting $\hat{T}_i^a < T_i^a, \hat{T}_i^s < T_i^s, \hat{T}_i^d > T_i^d$ is impossible because the limited misreports assumption we made earlier. Then, misreporting $\hat{t}_i < t_i$ cannot increase u_i because this only reduces the domain of $\tilde{t}_i$ and the allocation function allocates $\tilde{t}_i$ to maximise u_i (condition two of the above theorem). Finally, misreporting $\hat{a}_{i,r} < a_{i,r}$, $r \in R$ or $\hat{\Gamma}_l^i < \Gamma_l^i$ $l \in L$ will lead to the failure of the task, which reduces u_i to negative. Hence, agent i has no incentive to submit a non-truthful bid (i.e., where $(\hat{T}_i^a, \hat{T}_i^s, \hat{T}_i^d, \hat{v}_i, \{\hat{a}_{i,r}\}_{r \in R}, \{\hat{\Gamma}_l^i\}_{l \in L}) \neq (T_i^a, T_i^s, T_i^d, v_i, \{a_{i,r}\}_{r \in R}, \{\Gamma_l^i\}_{l \in L})$).

Then, we show that the conditions in Theorem 3.8 are also necessary. We first assume to the contrary that the first condition does not hold, i.e., the payment function is not independent of $\hat{v}_i$ or the payment function is not *monotonic*. In the former case, the mechanism is not DSIC according to Lemma 3.7. In the latter case, that is, there is some $T_i^{a\prime} < T_i^{a\prime\prime}$ such that the resource allocation is the same $(\lambda' = \lambda'')$ but the payment satisfies $p_i(\lambda', T_i^{a\prime}) > p_i(\lambda'', T_i^{a\prime\prime})$, while $\hat{T}_i^s$ and $\hat{T}_i^d$ remain unchanged. On this occasion, an agent whose true arrival time is $T_i^{a\prime}$ and who gets allocation λ' when reporting truthfully is incentivised to misreport $T_i^{a\prime}$ as $T_i^{a\prime\prime}$ because it can get the same allocation with less payment. Since this is contrary to the definition of DSIC, the first condition must be necessary.

Then, we assume that the first condition holds true but not the second. For instance, for some agent i with $v_i = v_i'$, there exists v_i'', the mechanism allocates λ_i' and λ_i''

respectively such that $v_i'(\tilde{t}_i') - p_i(\lambda_i', \hat{\theta}^{\langle \hat{T}_i^a \rangle}) < v_i''(\tilde{t}_i'') - p_i(\lambda_i'', \hat{\theta}^{\langle \hat{T}_i^a \rangle})$. On this occasion, this agent is incentivised to misreport v_i' as v_i''. Hence, the second condition is also necessary.

Finally, a mechanism that satisfies the Definition 3.5 is also IR for the following reasons. Since agent i will always bid truthfully (i.e., $\hat{v}(t) = v(t)$), the final allocation actually maximises the utility of agent i: $u_i = v_i(\tilde{t}_i) - p_i$. In addition, the maximum of u_i should be greater or equal to zero as i can always get a utility of zero with no resource allocated according to condition 3 in Definition 3.5. From the above discussion, it is clear that i will never get a negative utility under such mechanisms.

From the above, Theorem 3.8 is proven, and so is Theorem 3.6. □

3.3.2 Flexible Online Greedy (FlexOG) Mechanism

While OG is truthful, it is inefficient. This is because it does not change the allocation plans for arrived tasks, which may have a detrimental effect on the allocation of future high-value tasks. Our mechanism, FlexOG, extends OG by assigning newly arrived tasks greedily while maintaining allocation plans flexible. In greater detail, FlexOG employs a technique known as pre-commitment, in which FlexOG commits only the usage time associated with a task. This preserves the DSIC attribute of OG while increasing its flexibility. Additionally, this leads to increased societal welfare, as there is more room for optimisation in the future when high-value tasks arise. Algorithm 6 summarises FlexOG. After getting the report for task i, FlexOG performs the following procedures to determine the pricing for the t' amount of usage time. First, it finds the optimal allocation of all flexible tasks within the constraints of their committed usage time and t' assuming that task i is reported before k previous tasks. k starts from zero and ends to the number of tasks that arrives after $\hat{T}_i^a - t_{cap}$ except for task i. FlexOG computes the marginal operational cost, which is the difference between the total operational cost before and after task i arrives. Then the price for t' number of usage time is the maximum marginal operational cost for all k. Then, FlexOG selects t' that provides the most utility to task i as the committed usage time for task i, which guarantees that task i will receive t' usage time prior to its reported deadline $\hat{T}_i^d$. Following that, FlexOG asks for payment for task I, which was previously determined, and adds task i to the list of flexible tasks. Additionally, at the end of each time slot, FlexOG allocates resources for the subsequent time step based on the most recent allocation plans, which are those that maximise the social welfare of all flexible tasks given their committed usage time. Finally, if a task receives all of its committed usage time in the following time step, it is removed from the list of flexible tasks. In summary, our mechanism's key idea is that FlexOG commits only the usage time $\tilde{t}_i$ to task i but leaves the task's allocation plan flexible.

Theorem 3.9. *The FlexOG mechanism is DSIC, IR and WBB.*

Algorithm 6: The FlexOG mechanism

```
$\Theta_{arrived} \leftarrow [\,]$    // The list of arrived tasks
$\theta_{flex} \leftarrow [\,]$    // The list of flexible tasks
$\tilde{T} \leftarrow \{\,\}$    // The dictionary of (remaining) committed usage times of each task
$\Lambda \leftarrow \{\,\}$    // The dictionary of allocation decisions for each task
$t_{cap}$    // Twice the upper limit of the interval between the arrive time and the deadline of any
    task: $2 \times max\{(T_i^d - T_i^a)\}_{i \in I}$
for $t$ in $T$ do
    while new tasks arrive within $t$ do
        when a new task $i$ arrives    // Tasks arrive over time
        $\Theta_{arrived}.append(i)$    // Update the list of arrived tasks
        $\theta_{flex}.append(i)$    // Update the list of flexible tasks
        $\theta \leftarrow [\,]$    // The list of tasks that arrived between $[\hat{T}_i^a - t_{cap}, \hat{T}_i^a]$
        for $j$ in $\theta_{flex}$    // Loop through all tasks in $\theta_{flex}$
         do
            if $\hat{T}_j^a$ in $[\hat{T}_i^a - t_{cap}, \hat{T}_i^a]$    // If task $j$ arrived between $[\hat{T}_i^a - t_{cap}, \hat{T}_i^a]$
             then
                $\theta$.append($j$)    // Append that task to $\theta$
            end if
        end for
        $k \leftarrow 0$    // The number of tasks that $i$ is assumed to report before
        $\theta'_{flex} \leftarrow \theta_{flex}$    // The list of flexible tasks if $i$ is reported before $k$ previous tasks
        do
            Solve the maximum utility allocation for tasks in $\theta'_{flex} \setminus \{i\}$) (i.e.,
             $\arg\max_{\lambda_{j,k}} \sum_{j \in (\theta'_{flex} \setminus \{i\})} (\hat{v}_j(\lambda_{j,k}) - o_j(\lambda_{j,k}))$, given $\tilde{T}$)    // Find the allocation for $(\theta'_{flex} \setminus \{i\})$
             that maximise their social welfare, given their committed usage time
            for $t'$ in $[1, \hat{t}_i]$ do
                Solve the maximum utility allocation for tasks in $\theta'_{flex}$ (i.e.,
                 $\arg\max_{\lambda_{j,k,t'}} \sum_{j \in \theta'_{flex}} (\hat{v}_j(\lambda_{j,k,t'}) - o_j(\lambda_{j,k,t'}))$, given $\tilde{T}$ and $t'$)    // Find the allocation for $\theta'_{flex}$
                 that maximise their social welfare, given their committed usage time and the
                 assumed usage time of task $i$: $t'$
                if a solution is found then
                    $p_{i,k,t'} \leftarrow \sum_{j \in \theta'_{flex}} o_j(\lambda_{j,k,t'}) - \sum_{j \in (\theta'_{flex} \setminus \{i\})} o_j(\lambda_{j,k})$    // The payment for task $i$ of this $k$
                     and $t'$ is the corresponding marginal operational cost of task $i$
                else
                    $p_{i,k,t'} \leftarrow \infty$    // The payment for task $i$ of this $k$ and $t'$ is infinity
                end if
            end for
            $k \leftarrow k + 1$    // Increase $k$ by one
            $\theta$.pop$(-2)$    // Remove second to the last element from $\theta$
            $\theta'_{flex}$.pop$(-2)$    // Remove second to the last element from $\theta'_{flex}$
        while $\theta \neq [i]$
        for $t'$ in $[1, \hat{t}_i]$ do
            $p_{i,t'} \leftarrow \max_k(p_{i,k,t'})$    // The payment of $t'$ usage time is the maximum payment for all $k$
        end for
        $\tilde{t}_i \leftarrow \arg\max_{t'}(\hat{v}_i(t') - p_{i,t'})$    // The committed usage time for task $i$ is the usage time that
         maximise $i$'s utility
        $\tilde{T}[i] \leftarrow \tilde{t}_i$    // Commit task $i$'s usage time
        $p_i \leftarrow p_{i,\tilde{t}_i} = \max_k(p_{i,k,\tilde{t}_i})$    // Decide the payment for task $i$
    end while
    Solve the maximum utility allocation for tasks in $\theta_{flex}$ (i.e., $\arg\max_{\lambda_j} \sum_{j \in \theta_{flex}} (\hat{v}_j(\lambda_j) - o_j(\lambda_j))$, given $\tilde{T}$)    // Find
     the allocation for tasks in $\theta_{flex}$ that maximise their social welfare, given their
     committed usage time
    for $i$ in $\theta_{flex}$ do
        if the next time step $(t+1)$ is allocated to $i$ according to $\lambda_i$ then
            $\Lambda[i] \leftarrow \Lambda[i] \cup \lambda_{i,(t+1)}$    // Save the allocation decision of $(t+1)$ for task i
            $\tilde{T}[i] \leftarrow \tilde{T}[i] - 1$    // Update the remaining usage time of task $i$
        end if
        if $\tilde{T}[i] = 0$
         then
            $\theta_{flex} \leftarrow \theta_{flex} \setminus \{i\}$    // Delete task $i$ from the list of flexible tasks
        end if
    end for
end for
```

Proof. Following a similar argument, the FlexOG mechanism satisfies condition three in Definition 3.5. The payment of t' usage time: $p_{i,t'}$, which equals the maximum marginal operational cost: $\max_k (p_{i,k,t'})$ for all the cases if task i is reported before any task that arrive between $\left[\hat{T}_i^a - t_{cap}, \hat{T}_i^a\right]$, is monotonic and independent of $\hat{v}_i$ for the following reasons. In each case, the total operational cost is actually the lowest total operational cost that can still satisfy the committed usage time of every task. So it is independent of $\hat{v}_i$, and hence the marginal operational cost is also independent of $\hat{v}_i$. Moreover, increasing $\hat{T}_i^s$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$, $\{\hat{\Gamma}_l^i\}_{l \in L}$ or decreasing $\hat{T}_i^d$ can only increase the entire cost of operation $\sum_{j \in \theta'_{flex}} o_j(\lambda_{j,k,t'})$ as well as the marginal operational cost ($\sum_{j \in \theta'_{flex}} o_j(\lambda_{j,k,t'}) - \sum_{j \in \theta'_{flex} \setminus \{i\}} o_j(\lambda_{j,k})$) in the case that task i is reported before k tasks, following a similar argument in Theorem 4's proof. Since $p_{i,t'}$ is the maximum marginal operational cost of all k, so misreporting $\hat{T}_i^s$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$, $\{\hat{\Gamma}_l^i\}_{l \in L}$ higher or $\hat{T}_i^d$ lower can only increase $p_{i,t'}$. Finally, misreporting a higher $\hat{T}_i^a$ also can only increase the payment of t' usage time: $p_{i,t'}$ for the following reasons. Since agent i will not misreport $\hat{T}_i^a > T_i^d$, $T_i^a < \hat{T}_i^a \le T_i^d$. If i reports truthfully, then FlexOG considers $k \in [k_1, k_2]$ for $T_{(i-k)} \in \left[T_i^a - 2t_{cap}, T_i^a\right]$. Otherwise, FlexOG considers $k \in [k_3, k_4]$ for $T_{(i-k)} \in \left[\hat{T}_i^a - 2t_{cap}, \hat{T}_i^a\right]$. First, $p_{i,k_1,t'} = p_{i,k_3,t'}$ equals the lowest possible operational cost of t' usage time for task i because the time window of task $(i - k_3)$ and tasks reported before it will not overlap with the time window of task i. Second, $k_3 \ge k_4$ because $\hat{T}_i^a > T_i^a$. Hence the payment for t' usage time when i reports T_i^a truthfully: $p_{i,t'} = \max_{k \in [k_1,k_2]} (p_{i,k,t'})$ is less or equal to the payment when i misreports T_i^a: $\hat{p}_{i,t'} = \max_{k \in [k_3,k_4]} (p_{i,k,t'})$. Above all, this mechanism satisfies condition 1 in Definition 3.5 too.

Condition two is also met by the mechanism because it chooses the usage time t' that maximises $(\hat{v}_i(t') - p_{i,t'})$. From the above, the FlexOG mechanism is DSIC and IR by Theorem 3.6. Finally, because task i pays the maximum marginal operational cost, this payment will not be less than the actual marginal operational cost for task i. Hence, the total payment will not be less than the total operational cost. Thus, FlexOG satisfies WBB. □

3.3.3 Semi-Flexible Online Greedy (Semi-FlexOG) Mechanism

Although FlexOG is truthful and efficient in terms of social welfare, it is not very efficient in terms of processing time. This is because it needs to solve an NP-hard MILP problem whenever it needs to find the best allocation of all flexible tasks. To increase its scalability, we propose another mechanism called Semi-FlexOG, which is polynomial-time solvable. Similar to FlexOG, Semi-FlexOG commits only the usage time to a task, leaving its allocation plan flexible. However, in Semi-FlexOG, the allocation plans are not completely flexible as in FlexOG. The details of Semi-FlexOG is summarised in Algorithm 7. Upon receiving the report of task i, Semi-FlexOG finds the price for t' number of time steps by the following approach. First, Semi-FlexOG finds the payment for t' time steps for each start time $T' \in [\hat{T}_i^a, \hat{T}_i^d]$ that i is able to report and for the cases that i is reported before $j \in [0, \alpha]$ previous tasks by allocating tasks sequentially using Online Greedy mechanism and compute the corresponding marginal operational cost of task i as the payment. If it is impossible to satisfy all the committed usage time of the arrived tasks, then the payment for t', in this case, is infinity. α is called the level of flexibility here because it decides the upper bound of the number of previous tasks that task i is assumed to report before. Then, the payment for t' is the lowest payment of all the cases. After finding the payment for all possible numbers of usage time, Semi-FlexOG will choose the t' that gives task i the most utility as the committed usage time: $\tilde{t}_i$ for task i, and record the corresponding resource allocation plan. The payment for task i is the payment for $\tilde{t}_i$ decided previously. Similarly, at the end of each time slot, Semi-FlexOG allocates resources for the next time slot based on the most recent resource allocation plan, and if a task receives a time step in the following time slot, its remaining usage time is lowered by one.

Theorem 3.10. *The Semi-FlexOG mechanism is DSIC, IR and WBB.*

Proof. Under Semi-FlexOG mechanism, the payment for zero usage time is zero because the marginal operational cost is zero if the usage time is zero. Hence, Semi-FlexOG satisfies condition three of Definition 3.5. Furthermore, the payment for t' usage time is monotonic and independent of $\hat{v}_i$ for the following reasons. In each case, the payment is decided by the lowest operational cost that can satisfy all the resource requirements. So it is independent of $\hat{v}_i$, and the payment for t' usage time is the lowest payment of all cases, which is thus independent of $\hat{v}_i$. Following a similar argument in Theorem 4's proof, the payment for t' usage time: $p_{i,t'}$ increases over $\hat{T}_i^a$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$, $\{\hat{\Gamma}_l^i\}_{l \in L}$. Furthermore, the payment also increases over $\hat{T}_i^s$ and decreases over $\hat{T}_i^d$ because increasing $\hat{T}_i^s$ or decreasing $\hat{T}_i^d$ can only reduce the space of total cases. Furthermore, Semi-FlexOG also satisfies condition two of Definition 3.5 because it chooses the allocation plan that maximises task $i's$ utility. Above all, the Semi-FlexOG mechanism is DSIC and IR by Theorem 3.6. Finally, because the payment of task i equals its marginal operational cost, the total payment will equal the total operational cost. Thus, Semi-FlexOG satisfies WBB as well. □

3.3.4 Truthfulness Without Limited Misreport of Deadlines

As we discussed before, agents can report later deadlines for some types of tasks when withholding the results until the reported deadline is not feasible. In that case, OG and SWMOA2 mechanisms are still DSIC, while FlexOG is not DSIC anymore. We first give an example showing why FlexOG is not DSIC. For example, suppose agent i reports a later deadline $\hat{T}_i^d$ ($\hat{T}_i^d > T_i^d$) and reports other information of its task truthfully. It

Algorithm 7: The Semi-FlexOG mechanism

```
$\Theta_{arrived} \leftarrow [\,]$    // The list of arrived tasks
$\tilde{T} \leftarrow \{\,\}$    // The dictionary of (remaining) committed usage times of each task
$\Lambda \leftarrow \{\,\}$    // The dictionary of allocation decisions for each task
$\alpha$ // The upper bound of the number of previous tasks that any task is assumed to report before
   (level of flexibility)
for $t$ in $T$ do
    while new tasks arrive within $t$ do
        when a new task $i$ arrives    // Tasks arrive over time
        $\Theta_{arrived}.append(i)$    // Update the list of arrived tasks
        for $t'$ in $[1, \hat{t}_i]$ do
            for $T'$ in $[\hat{T}_i^a, \hat{T}_i^d]$ do
                for $j$ in $[0, \alpha]$ do
                    $\Lambda' \leftarrow \{key : \Lambda[key]$ for key in $[0, i-k-1]\}$    // The resource allocation decitions
                      for the tasks that are assumed to report before task $i$
                    solve the maximum utility allocation for task $i$ (i.e., $\arg\max_{\lambda_{i,j,t',T'}}(\hat{v}_i(t') - o(\lambda_{i,j,t',T'}))$, given $\Lambda'$
                      and $t'$)    // Find the allocation for $i$ that maximises its social welfare,
                      given existing allocation decitions and the assumed usage time of task $i$:
                      $t'$
                    if there is no solution then
                        $p_{i,j,t',T'} \leftarrow \infty$    // Payment for this setting is infinity
                    else
                        $\Lambda'[i] \leftarrow \lambda_{i,j,t',T'}$    // Save the allocation decision for task $i$
                        /* Make allocation decisions for the remaining tasks    */
                        for $k$ in $[i-k, i-1]$ do
                            solve the maximum utility allocation for task $k$ (i.e.,
                              $\arg\max_{\lambda_{k,j,t',T'}}(\hat{v}_k(\tilde{T}[k]) - o(\lambda_{k,j,t',T'}))$, given $\Lambda'$ and $\tilde{T}[k]$)    // Find the allocation
                              for $k$ that maximises its social welfare, given existing
                              allocation decitions and the committed usage time of task $k$: $\tilde{T}[k]$
                            if there is no solution then
                                $p_{i,j,t',T'} \leftarrow \infty$    // Payment for this setting is infinity
                            end if
                        end for
                        if $p_{i,j,t',T'} \neq \infty$ then
                            $p_{i,k,t'} \leftarrow \sum_{j \in \theta'_{flex}} o_j(\lambda_{j,k,t'}) - \sum_{j \in (\theta'_{flex} \setminus \{i\})} o_j(\lambda_{j,k})$    // The payment for task $i$ of
                              this $k$ and $t'$ is the corresponding marginal operational cost of
                              task $i$
                        end if
                    end if
                end for
            end for
        end for
        $\arg\max_{j,t',T'}(\hat{v}_i(t') - p_{i,j,t',T'})$    // Choose the setting that maximises task $i$'s utility
        $\tilde{t}_i \leftarrow t'$    // Commit task $i$'s usage time
        $\tilde{T}[i] \leftarrow \tilde{t}_i$    // Record task $i$'s committed usage time
        $p_i \leftarrow p_{i,j,t',T'}$    // Decide the payment for task $i$
        for $l$ in $[i-k, i]$ do
            $\Lambda[l] \leftarrow \lambda_{l,j,t',T'}$    // Update the allocation decisions
        end for
    end while
end for
for $i$ in $\Theta_{arrived}$ do
    if the next time step $(t+1)$ is allocated to $i$ according to $\Lambda$ then
        $\tilde{T}[i] \leftarrow \tilde{T}[i] - 1$    // Update the remaining usage time of task $i$
    end if
end for
```

may get a lower payment because the payment function p_i is monotonic according to Definition 3.5. Then, it is possible that its committed time steps get rescheduled in its time window ($[T_i^s, T_i^d]$) because of tasks that arrive in the future. In that case, agent i gets more utility by misreporting a later deadline because it gets the same value with a lower payment. However, if we modify FlexOG so that the allocated time steps for tasks can only be rescheduled to later time steps, then FlexOG will still be DSIC in this case. The following theorems show that OG, SWMOA2 and modified FlexOG are still DSIC and IR when agents can report later deadlines for their tasks.

Theorem 3.11. *The OG and SWMOA2 mechanisms are DSIC and IR when agents are able to report their deadlines arbitrarily.*

Proof. First, an agent cannot increase its utility by reporting an earlier deadline or misreporting other information about its task (under the limited misreport assumptions in Section 3.1.2) according to Theorem 3.2 and 3.3. Then, we analyse the case where agent i reports a later deadline. Suppose the utility of agent i is u_i, and its payment function is p_i when it reports truthfully, and the utility of agent i is u_i', and its payment function is p_i' when it reports a later deadline. If the usage time is still allocated in its time window $[T_i^s, T_i^d]$, it gets the same utility ($u_i' = u_i$). If some time steps ($\tilde{t}_1$) are allocated in its time window, while some time steps ($\tilde{t}_2$) are not, then, $u_i' = v_i(\tilde{t}_1) - p_i'(\tilde{t}_1) - p_i'(\tilde{t}_2)$, and $u_i = v_i(\tilde{t}_1 + \tilde{t}_2) - p_i(\tilde{t}_1) - p_i(\tilde{t}_2)$. Since, ($\tilde{t}_1$) is inside the time window, $p_i'(\tilde{t}_1) = p_i(\tilde{t}_1)$, and $u_i' = v_i(\tilde{t}_1) - p_i(\tilde{t}_1) - p_i'(\tilde{t}_2)$. Now, $v_i(\tilde{t}_1 + \tilde{t}_2) - p_i(\tilde{t}_1) - p_i(\tilde{t}_2) \geq v_i(\tilde{t}_1) - p_i(\tilde{t}_1)$ because OG and SWMOA2 always choose the allocation that maximises agent i's utility. Thus, the utility of agent i when it reports a later deadline is also no more than its utility when it reports truthfully ($u_i' \leq u_i$). Therefore, the OG and SWMOA2 mechanisms are still DSIC when agents can report later deadlines. Furthermore, agents will get a utility of at least zero because the payment for no allocation is zero and the machanisms maximise the agents' utilities. Therefore, these mechanisms also satisfy IR. □

Theorem 3.12. *The modified FlexOG mechanism is DSIC and IR when agents are able to report their deadlines arbitrarily.*

Proof. Similar to the proof above. An agent cannot increase its utility by reporting an earlier deadline or misreporting other information about its task (under the limited misreport assumptions in Section 3.1.2) according to Theorem 3.9. Then, we analyse the case where agent i reports a later deadline. Suppose the utility of agent i is u_i, and its payment function is p_i when it reports truthfully, and the utility of agent i is u_i', and its payment function is p_i' when it reports a later deadline. If in the end, the usage time is still allocated in its time window $[T_i^s, T_i^d]$, it gets the same utility ($u_i' = u_i$). This is because the initial allocation must also be in its time window, since the modified FlexOG can only reschedule its usage time to later time steps. If in the end, some time steps ($\tilde{t}_1$) are allocated in its time window, while some time steps ($\tilde{t}_2$) are not, then, $u'_i = v_i(\tilde{t}_1) - p'_i(\tilde{t}_1) - p'_i(\tilde{t}_2)$, and $u_i = v_i(\tilde{t}_1 + \tilde{t}_2) - p_i(\tilde{t}_1) - p_i(\tilde{t}_2)$. Since, ($\tilde{t}_1$) is inside the time window, $p'_i(\tilde{t}_1) = p_i(\tilde{t}_1)$, and $u'_i = v_i(\tilde{t}_1) - p_i(\tilde{t}_1) - p'_i(\tilde{t}_2)$. Now, $v_i(\tilde{t}_1 + \tilde{t}_2) - p_i(\tilde{t}_1) - p_i(\tilde{t}_2) \geq v_i(\tilde{t}_1) - p_i(\tilde{t}_1)$ because FlexOG always choose the allocation that maximises agent i's utility. Thus, the utility of agent i when it reports a later deadline is also no more than its utility when it reports truthfully ($u'_i \leq u_i$). Therefore, the modifield FlexOG is DSIC when agents can report later deadlines. Furthermore, this mechanisms also satisfy IR because agents get at least zero utility by getting zero usage time. $\square$

3.4 Simulations and Results

We evaluate our proposed mechanism FlexOG and Semi-FlexOG in this section using extensive simulations. First, we describe how the synthetic data is generated. We employ synthetic data in our simulations because no comprehensive dataset of real-world fog computing tasks available at the moment. Following that, we assess FlexOG's social welfare performance under various configurations of the synthetic data.

3.4.1 Experimental Setup

In order to simulate a small fog computing network, we choose the following parameters. Our discrete time period has a duration of $|T| = 12$. Six fog nodes ($|P|$=6) and six locations ($|L|$=6) are owned by the fog provider. Three classic topologies of the network are used in our simulations: an almost fully connected topology, a ring topology and a line topology (see Figures 3.2,3.3 and 3.4). Additionally, each fog node has $|R| = 3$ types of computational resources (CPU, RAM, and storage). This configuration was chosen to represent the following tiny community's fog computing network. There are six houses in this community. Each house is a location, and each house is connected to a fog node. One time step corresponds to one hour in real life, and the whole time span corresponds to a day. Another reason we chose this tiny setting is that it allows us to execute more trials for all algorithms in a reasonable amount of time.

This time period contains $|I| = 40$ tasks. The arrival time Tia follows a continuous uniform distribution $U(0, 10)$, which implies that no two tasks arrive at the same time, which was an assumption stated before in our RAFC model. Furthermore, the number of endpoints for each task Ei is uniformly generated from $\{1, 2, \ldots, 6\}$, and the location of each endpoint $u^i_{e,l}$ is uniformly determined at random from all L locations with replacement.

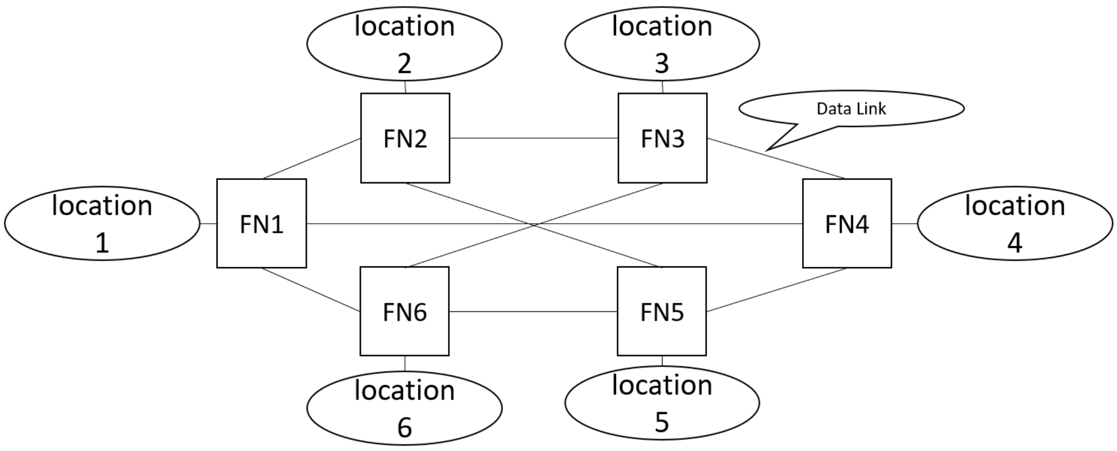

FIGURE 3.2: The (almost fully connected) topology of the fog computing system.

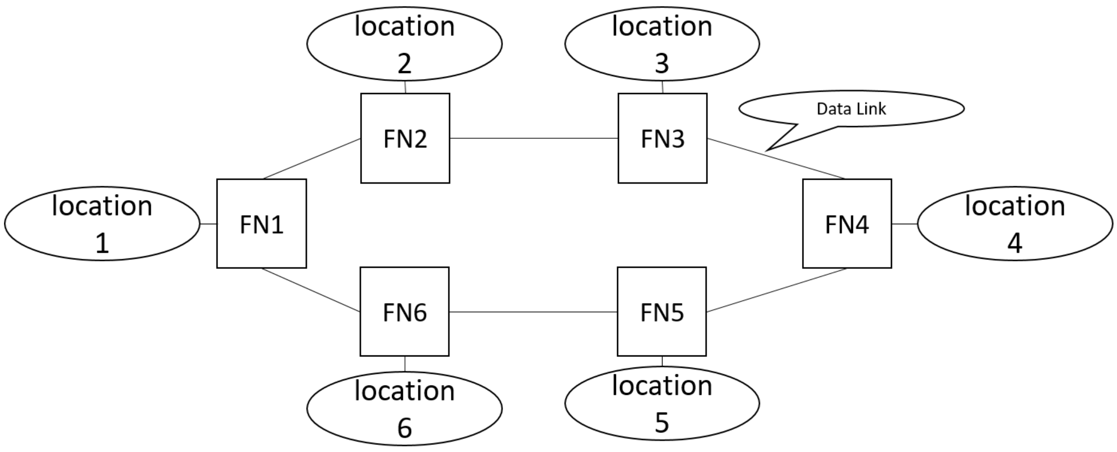

FIGURE 3.3: The (ring) topology of the fog computing system.

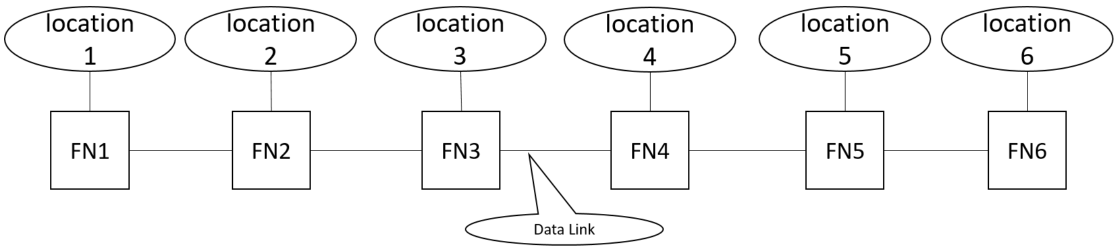

FIGURE 3.4: The (line) topology of the fog computing system.

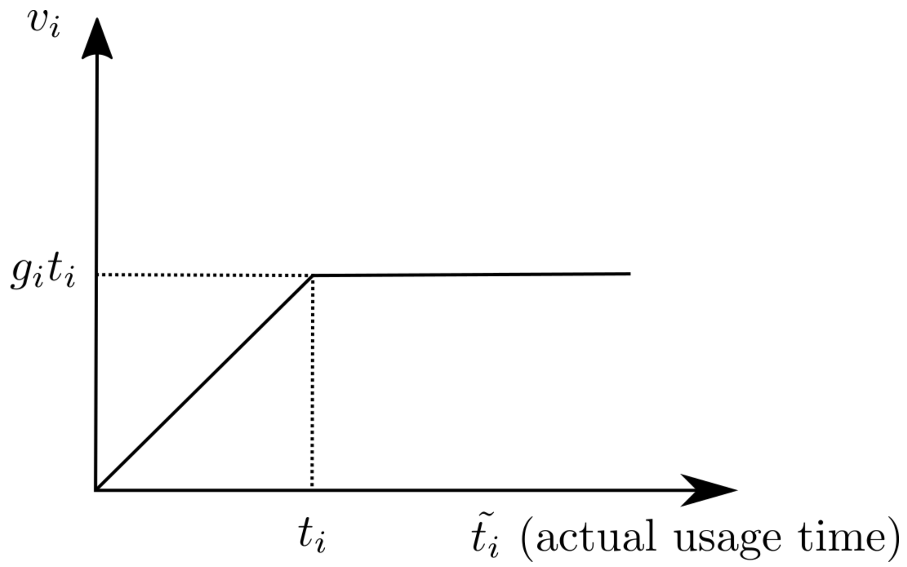

FIGURE 3.5: The valuation function of task *i*.

Moreover, in our simulations, we use a special valuation function v_i that is a non-decreasing linear function of *i*'s usage time $\tilde{t}_i$:

$$v_i(\tilde{t}_i) = \begin{cases} g_i \times \tilde{t}_i & \text{if } \tilde{t}_i \le t_i \\ g_i \times t_i & \text{if } \tilde{t}_i > t_i \end{cases}$$

Where the valuation coefficient g_i represents task *i*'s obtained value per usage time. An example of such a valuation function is shown in Figure 3.5.

Since the value densities (i.e., the average valuation of each time step) of fog tasks vary considerably in real life. To make resource allocation more realistic, this synthetic data contains two categories of tasks: low-value tasks and high-value tasks. $y \in [0,1]$ denotes the fraction of high-value tasks. For either type of task $i:$ and $\Gamma_l^i\ \forall l \in L$ are all generated using a Gaussian distribution $\mathcal{N}(1,1)$ with negative values removed. The duration of usage time t_i is a positive integer uniformly chosen from the range $\{1,2,3,4\}$, and the start time T_i^s is an integer uniformly chosen within two time steps following the arrival time: $\{\lceil T_i^a \rceil, \lceil T_i^a \rceil + 1, \lceil T_i^a \rceil + 2\}$. Additionally, the deadline T_i^d is an integer that is uniformly picked between b and d time steps following the earliest finish time (but not greater than the final time step): $\{T_i^s + t_i - 1 + b, T_i^s + t_i + b, \ldots, min(T_i^s + t_i - 1 + d, |T|)\}$. Thus, (b,d) specifies the task's deadline slackness, a critical parameter that shows the task's flexibility. g_i is uniformly picked from the continuous interval: $[8,30]$ for a low-value task i . For a high-value task i , on the other hand, g_i is uniformly chosen from the continuous interval: $[180,200]$. Thus, the upper bound of the ratio of the highest and lowest valuation coefficients in this example is $F = \frac{200}{8} = 25$

Finally, the aggregate resource capacity of each resource r is calculated as follows: The

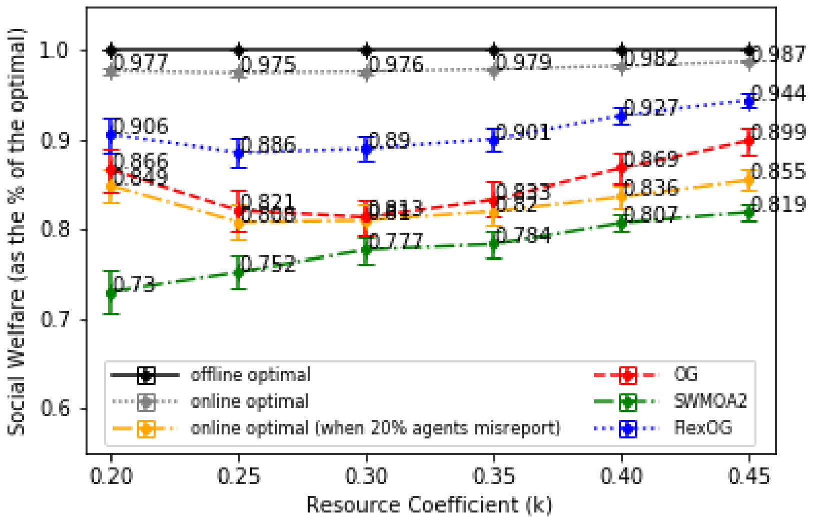

FIGURE 3.6: The social welfare achieved by four mechanisms with parameters $((b, d) = (5, 10), F = 25, y = 0.1)$ and almost fully connected network.

value of $\sum_{w \in W} A_{w,r}$ is set to be a k proportion of the corresponding total resource demand of $\sum_{i \in I} a_{i,r}$, as well as the total bandwidth capacity: $\sum_{(j,k) \in \mathbb{E}} b_{j,k}$ is set to a $2k$ percentage of the overall bandwidth requirements $\sum_{i \in I, l \in L} \Gamma_l^i$ because data traffic is typically distributed across many data links. Then, each fog node receives the same fraction of the resource r: $\frac{\sum_{w \in W} A_{w,r}}{|W|}$, and each data link receives the same fraction of the total bandwidth available: $\frac{\sum_{(j,k) \in \mathbb{E}} b_{j,k}}{|\mathbb{E}|}$. Thus, parameter k reflects the scarcity of resources in the fog and is referred to in this thesis as the resource coefficient. Finally, the cost per unit of resource at various fog nodes and links: $o_{w,r}\ w \in W, r \in R$ and $o_{j,k}, (j,k) \in \mathbb{E}$ are all uniformly generated from [0.03, 0.1].

3.4.2 Performance of FlexOG

We evaluated our mechanism's robustness by running simulations with a variety of parameters, including network topologies, resource scarcity in fog nodes and data links, and task deadline slackness. The trends are consistent across all of these situations. FlexOG's performance in social welfare is typically around 90% of the theoretical upper bound and between 5% and 10% better than that of OG. In the following, we will show the results of these simulations and analyse them.

3.4.2.1 Social Welfare for Different Levels of Resource Scarcity

First, we compare the total social welfare achieved by FlexOG to other benchmarks using different resource coefficients k to indicate the scarcity of resources in Figure 3.6,

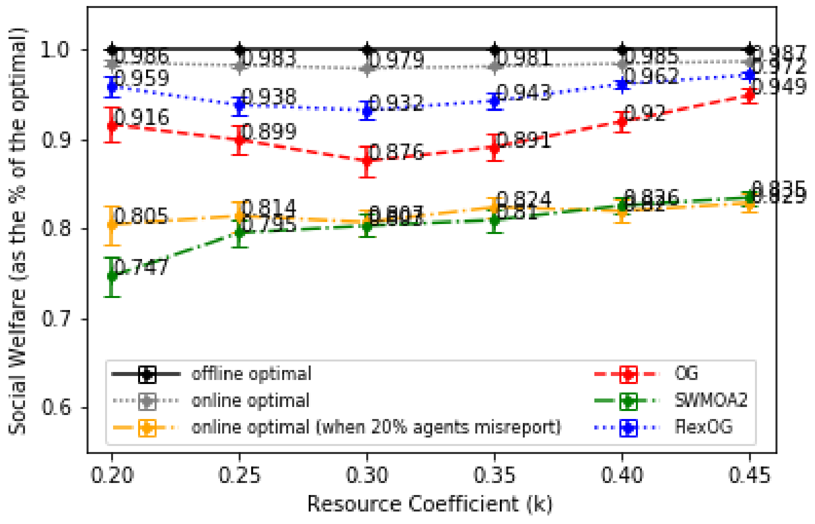

FIGURE 3.7: The social welfare achieved by four mechanisms with parameters $((b,d) = (5,10), F = 25, y = 0.1)$ and ring network.

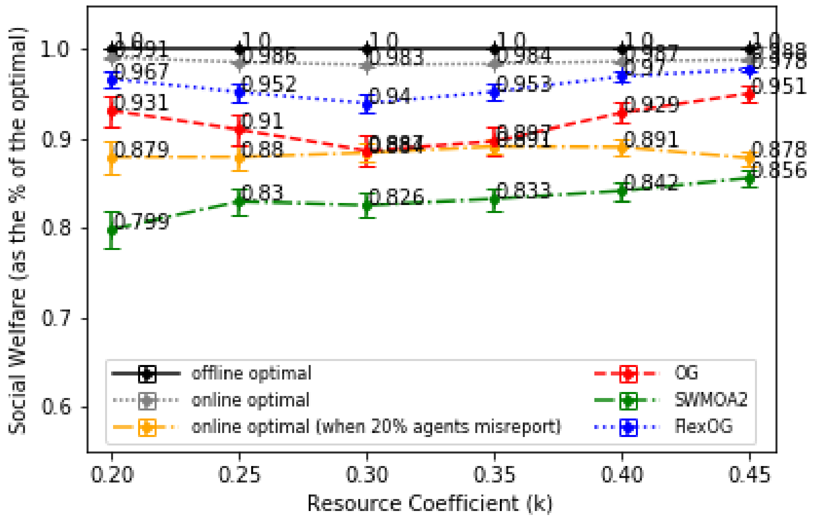

FIGURE 3.8: The social welfare achieved by four mechanisms with parameters $((b,d) = (5,10), F = 25, y = 0.1)$ and line network.

3.7 and 3.8[1]. Nota bene, we normalise the results to the offline optimal performance to make comparisons between different mechanisms easier. Since the trend of all three figures is similar, we analyse Figure 3.6 as a representative in the following.

Figure 3.6 demonstrates that FlexOG consistently produces more social welfare than alternative truthful benchmarks(i.e., OG and SWMOA2). SWMOA2 in particular always performs poorly, owing to the fact that its virtual prices are exponentially related to the load factors of the resource, preventing jobs from being allocated even when there is sufficient resource. This phenomenon gets even more significant when the resource is more scarce, i.e., when k is lower. For example, the average social welfare achieved by SWMOA2 is only 78.15% of that achieved by FlexOG when the resource coefficient $k = 0.2$, while for other higher k this number is around 87%. This is because, when the resource is more scarce, the load factors of resources increase faster, and so do the virtual prices.

Furthermore, the performance of FlexOG is about 5%-10% better than that of OG in terms of social welfare. This is because under FlexOG, when and how the committed time steps are allocated to tasks is flexible. Thus, FlexOG can reschedule unfinished tasks in order to free up more time steps for the newly arrived high-value task, whereas OG cannot. Additionally, the figure also demonstrates that the performance gap between FlexOG and OG narrows when the resource coefficient k is either small or large. Intuitively, this is because when resources are few or abundant, OG performance will be closer to optimal, leaving less room for FlexOG to increase social welfare through task rescheduling. This indicates that the superiority of FlexOG is more significant when resources are neither too scarce nor too abundant, which is in correspondence with reality in most cases.

In addition, our mechanism also performs close to offline optimal, achieving around 90% in the almost fully connected network and around 95% in other topologies, which indicates that our mechanism is efficient even though it is online. Interestingly, the performance of online optimal is very close to that of offline optimal. The main reason is that all tasks are preemptible, so myopic decisions will not have a significant effect in the future. If a high-value task arrives, online optimal can always pause some low-value tasks to process the newly arrived high-value one. However, online optimal is not truthful and vulnerable to manipulations. So, although online optimal performs about 10% better than FlexOG, its performance drops below that of FlexOG when just 20% of agents misreport. Here, agents who misreport only misreport their valuation coefficient higher (as 1 million), and report other information truthfully. So the performance of online optimal is mainly used as an indicator of the upper bound of

[1]All statistics are within a 95% confidence interval based on 200 trials, and the CPLEX optimiser's relative tolerance is set to 1% for offline optimum and 5% for other methods. (A 1% tolerance indicates that the optimiser will terminate when a solution is within 1% of optimality.) We establish the relative tolerance for offline optimal lower to improve the accuracy of the theoretical upper bound on social welfare.

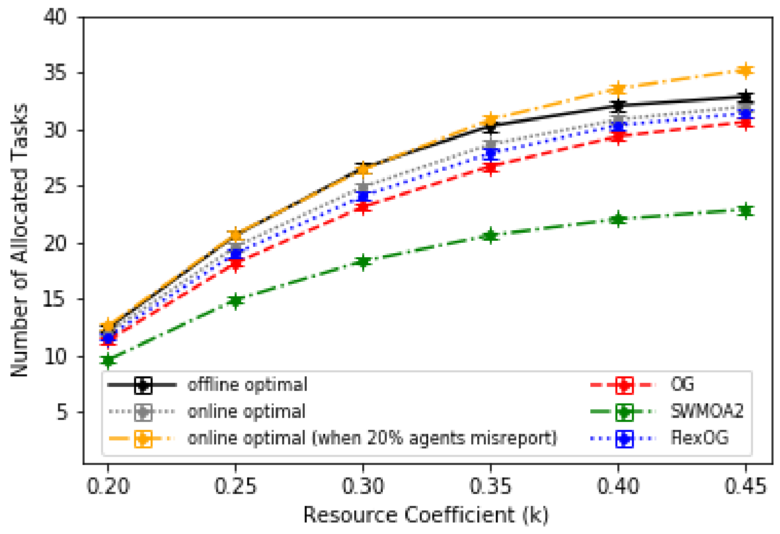

FIGURE 3.9: The number of tasks got allocated by five mechanisms with parameters $((b,d) = (5,10), F = 25, y = 0.1)$ and the almost fully connected network.

online resource allocation here. In addition, we have also tested whether agents have an incentive to misreport under the online optimal mechanism by comparing the utilities of truthful and non-truthful agents (see Section 3.4.2.4), and the result shows on average non-truthful agents get a higher utility. This means that, in a strategic setting where agents can misreport, FlexOG can actually achieve significantly more social welfare than online optimal.

Finally, Figure 3.9 shows the number of tasks that get allocated (i.e., tasks that get at least one time step usage time) under different mechanisms with the almost fully connected network. The result is similar for other network topologies, so these results are omitted in this thesis. We can see from the figure that resources are truly scarce when $k = 0.2$ because only around 30% of tasks get allocated under offline optimal. While the resources are relatively abundant when $k = 0.45$, where around 75% of tasks get allocated under offline optimal. In addition, FlexOG only allocates about one or two tasks more than OG on average, which means that FlexOG achieves more social welfare mainly by giving more usage time to high-value tasks rather than getting more tasks allocated. Interestingly, most tasks get allocated under online optimal when 20% of agents misreport. Intuitively, this is because some low-value tasks which would not get allocated under other mechanisms can get some usage time because they misreport their valuation coefficient higher. Additionally, because the virtual cost of SWMOA2 is frequently significantly greater than the operational cost, tasks are rejected even when sufficient resources are available to execute them, it has the lowest number of allocated tasks.

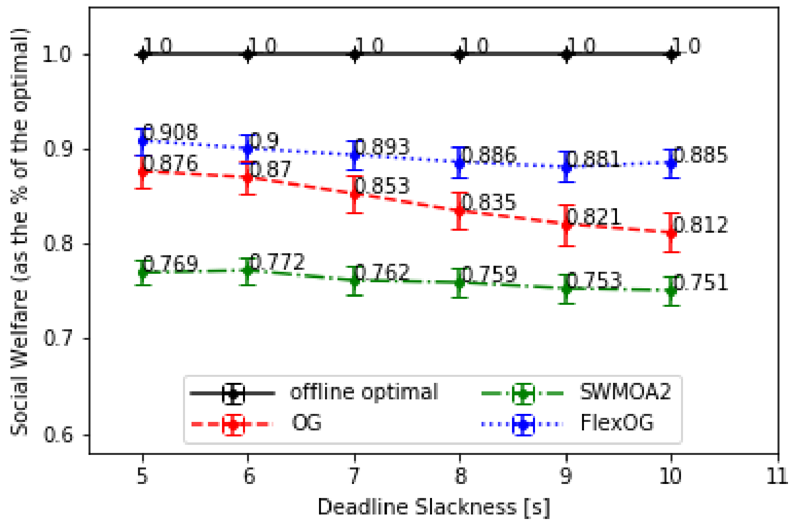

FIGURE 3.10: The social welfare achieved by four mechanisms with parameters $((b, d) = \{(0, 5), (1, 6), (2, 7)(3, 8), (4, 9), (5, 10)\}, F = 25, q = 0.1, k = 0.3)$ and almost fully connected network.

3.4.2.2 Social Welfare for Different Levels of Deadline Slackness

Now we compare the performance of various network topologies in terms of social welfare under varying levels of deadline slackness (Figures 3.10, 3.11 and 3.12). We do not include the online optimal algorithm here because it is not truthful, and we have already shown that its efficiency is lower than that of the OG and FlexOG mechanisms in a strategic setting. A task with a greater degree of deadline slack has more time steps between its earliest possible completion time and its deadlines and is therefore more flexible in terms of resource allocation. As illustrated in the figures, the difference between FlexOG and OG widens as task deadline slackness grows. This is because when tasks are more flexible, FlexOG is more likely to reschedule low-value tasks to make room for higher-value tasks, whereas OG cannot benefit from this because its resource allocation is inflexible. In addition, SWMOA2's performance is quite stable in terms of different levels of deadline slackness. This is because, instead of the flexibility of tasks, the virtual resource pricing has a significant impact on SWMOA2's performance.

3.4.2.3 Processing Time of Different Algorithms

In this section, we compare the processing time of FlexOG with the benchmark mechanisms. The processing time is also important because many agents cannot wait for a long time for an allocation decision, and the fog provider does not want to use too much computing resource on making allocation decisions. Thus, in this thesis, the

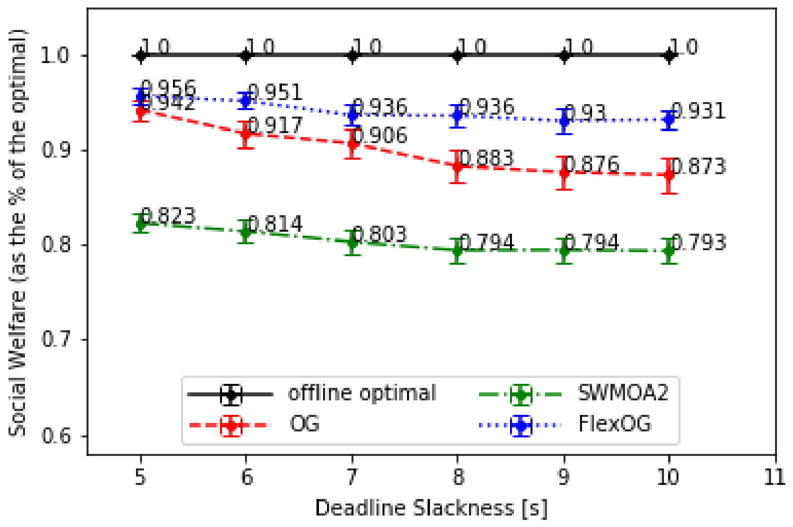

FIGURE 3.11: The social welfare achieved by four mechanisms with parameters $((b,d) = \{(0,5),(1,6),(2,7)(3,8),(4,9),(5,10)\}, F = 25, q = 0.1, k = 0.3)$ and ring network.

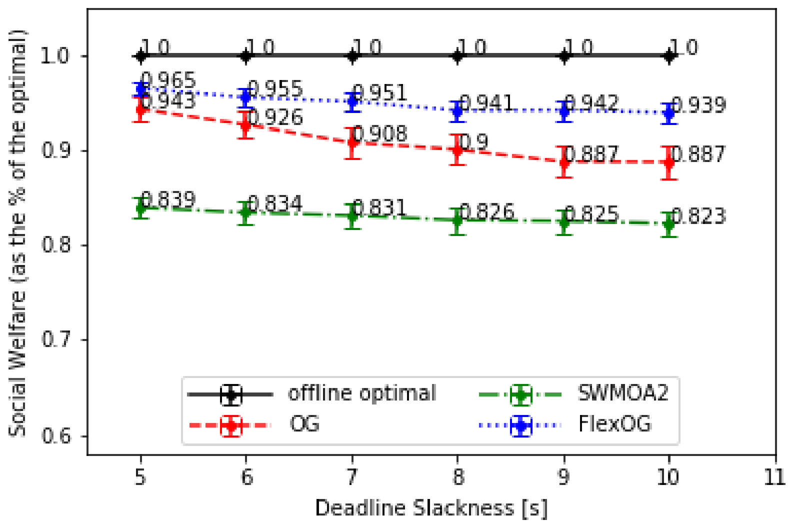

FIGURE 3.12: The social welfare achieved by four mechanisms with parameters $((b,d) = \{(0,5),(1,6),(2,7)(3,8),(4,9),(5,10)\}, F = 25, q = 0.1, k = 0.3)$ and line network.

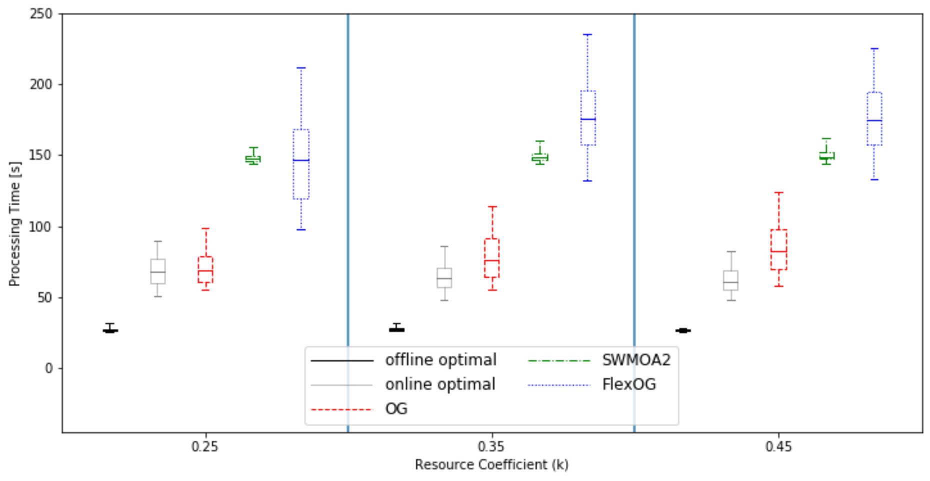

FIGURE 3.13: The processing time of five mechanisms with parameters $((b,d) = (5,10), F = 25, y = 0.1)$ and almost fully connected network.

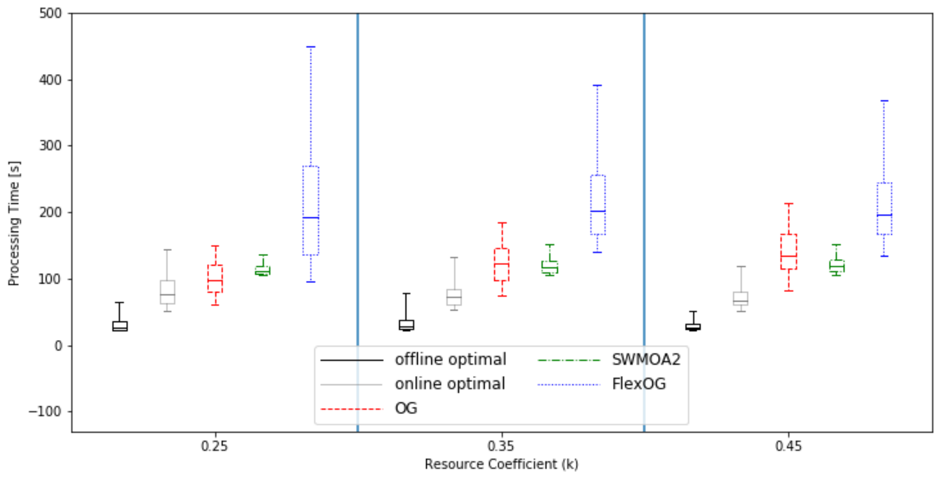

FIGURE 3.14: The processing time of four mechanisms with parameters $((b,d) = (5,10), F = 25, y = 0.1)$ and ring network.

shorter the processing time, the better (Challenge 8). The simulation was conducted on the Iridis 4 Compute Cluster[2], using four cores of an Intel Xeon E5-2670 ('Sandybridge') processor with 16GB RAM.

The results are shown in Figure 3.13, 3.14 and 3.15. We plot the processing time of all mechanisms under resource coefficients $k = 0.25$, $k = 0.35$ and $k = 0.45$. The processing time under different resource coefficients exhibits a similar pattern and is not plotted to make the figures easier to read. Note that the boxes reflect the bottom to upper 25 percentile values of the data with whiskers showing 5 to 95 percentile of the

[2]https://hpc.soton.ac.uk/redmine/projects/iridis-4-support/wiki

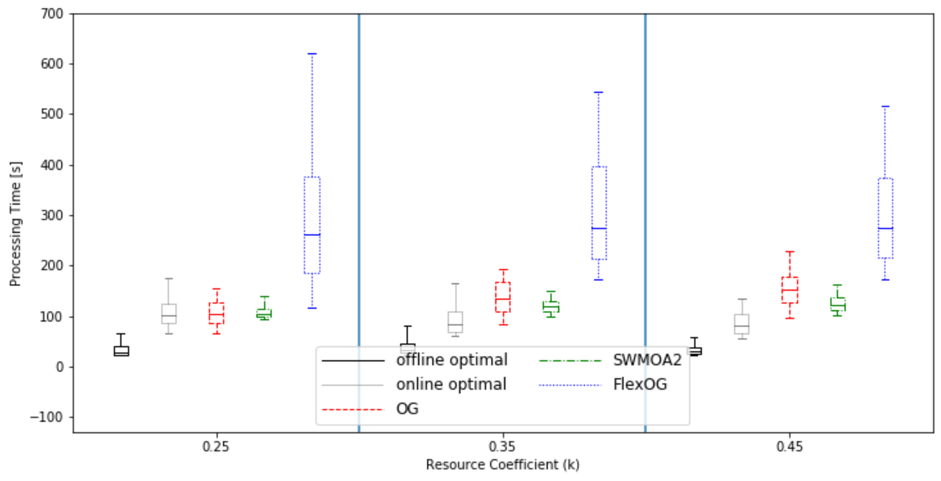

FIGURE 3.15: The processing time of four mechanisms with parameters $((b, d) = (5, 10), F = 25, y = 0.1)$ and line network.

data, and the outliers are discarded.

The figure demonstrates that, on average, offline optimal takes the least amount of time to process, online optimal and OG take more time to process, and SWMOA2 and FlexOG take the most time to process. This is mostly due to the fact that offline optimal only needs to solve the optimization problem once, whereas all other algorithms must solve it multiple times. Online optimal must solve the optimization problem at the beginning of each time step except the first, which means it must solve it $|T| - 1 = 11$ times. The remaining algorithms must solve the optimization problem each time a new task is encountered, which means they must solve it $|I| = 40$ times. However, the processing time of offline optimal is much more than $\frac{1}{11}$ of that of online optimal. This is because offline optimal needs to make a resource allocation decision for all tasks at once, while online optimal only needs to make a decision for tasks that arrived in the previous time step every time. So offline optimal's optimisation problems have more decision variables and take more time to solve on average. Another reason is that the relative tolerance of offline optimal is set to be 0.01 while this parameter is set at 0.05 for other algorithms, which also causes offline optimal to take more time to find a solution to its optimisation problem.

Similarly, although OG needs to solve the optimisation problem four times as many time as online optimal, it has a similar processing time on average. This is because OG only needs to make a resource allocation decision for just one task at a time. So, its optimisation problem is solved faster than online optimal's on average.

Finally, FlexOG uses the greatest time since it not only must solve the optimisation problem $|I| = 40$ several times but also because its optimization problems contain more decision variables. This is because it needs to make allocation decisions for all flexible

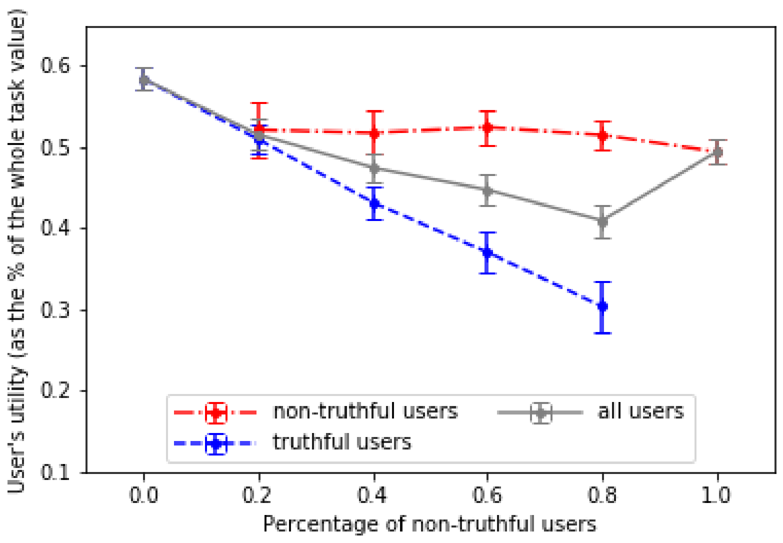

FIGURE 3.16: Utility comparison between truthful and non-truthful agents ($(b, d) = (5, 10), F = 25, y = 0.1$) in almost fully connected network.

tasks each time a new task arrives. The reason why SWMOA2 takes the second-highest time on average is that SWMOA2 needs to compute the load factor and virtual price of each resource after every allocation decision, while other algorithms just use the fixed operational cost of each resource. Interestingly, the processing time of FlexOG also has a wider dispersion than other algorithms. This is because with more optimisation problems to solve and with more decision variables, FlexOG has a higher probability of coming across optimisation problems that are hard to solve. For each task, FlexOG takes around four seconds to make the allocation decision on average. For tasks whose arrival time is much earlier than their start time, it is feasible to use FlexOG because there is sufficient time for FlexOG to make allocation decisions. However, if the arrival time and the start time of a task are the same, or tasks arrive frequently, then the processing time of FlexOG will become an issue. Furthermore, our simulation is based on a small fog network, and FlexOG will take even more time to make decisions for larger fog networks. We will discuss how we plan to solve this issue in the future work section (Section 5.2).

3.4.2.4 Utility Comparision between Truthful and Non-truthful Agents

In this section, we examine whether IoT users are incentivised to report their tasks truthfully by comparing the average utility of non-truthful and truthful agents. The result of what percentage of the total value of tasks (i.e., the sum of value for IoT users if their tasks are all fully accomplished) are achieved by all agents, truthful agents and non-truthful agents respectively under the online optimal algorithm in the almost fully connected topology is shown in Figure 3.16. Recall that non-truthful agents always report their valuation coefficient as one million. This is because if they can achieve a higher utility compared to truthful agents by misreporting their valuation coefficient, they will have an even bigger advantage when they misreport other information as well. Although this may not be the optimal strategy for non-truthful agents, it is enough to show the impact of non-truthful agents. The figure shows that non-truthful agents have similar utility as truthful agents when 20% of IoT users are non-truthful. However, the utilities of truthful agents decrease rapidly as more agents become non-truthful, and non-truthful agents have about a 67% higher utility than truthful agents on average, when 80% of agents are non-truthful. This is because the reported valuation coefficients of non-truthful agents are much higher than that of truthful agents, and online optimal will always prioritise non-truthful agents when it makes allocation decisions. This means that under online optimal, IoT users have no incentives to be truthful. Thus, the social welfare achieved by online optimal when all agents report their types truthfully cannot be achieved in a strategic setting.

Figure 3.16 also indicates that non-truthful agents have a negative impact on the overall utility (social welfare). In more detail, the total social welfare decreases rapidly as more agents become non-truthful. This is due to the fact that online optimal makes allocation decisions based on false valuation coefficients, so it may prioritise a low-value task over a high-value task. However, when all agents are non-truthful, online optimal achieves a higher social welfare than that when 80% agents are non-truthful. The reason for this is that when all agents are non-truthful, their valuation coefficients are all the same, the case where a low-value task is prioritised over a high-value task because of its misreported valuation coefficient disappears. The main reason why social welfare is lower when all agents are non-truthful than that when all agents are truthful is that low-value tasks are still prioritised when all agents are non-truthful because they need fewer resources on average.

3.4.3 Performance of Semi-FlexOG

3.4.3.1 Social Welfare for Different Levels of Resource Scarcity

Figure 3.17 shows the social welfare achieve by five algorithms (100 trials). As shown in the figure, FlexOG and Semi-FlexOG's performance in social welfare is better than other truthful benchmarks', and. Furthermore, FlexOG and Semi-FlexOG's performance in social welfare is better than other non-truthful algorithms' when some (from a small proportion to all) agents misreport their tasks. Here, the total number of tasks is 40, the deadline slackness of high-value tasks is zero, and the deadline slackness of low-value tasks is six. For Semi-FlexOG, we consider the cases where a task is assumed to report at most before two, three or five previous tasks. For FlexOG, we consider the price assuming a task is reported at most before three previous tasks.

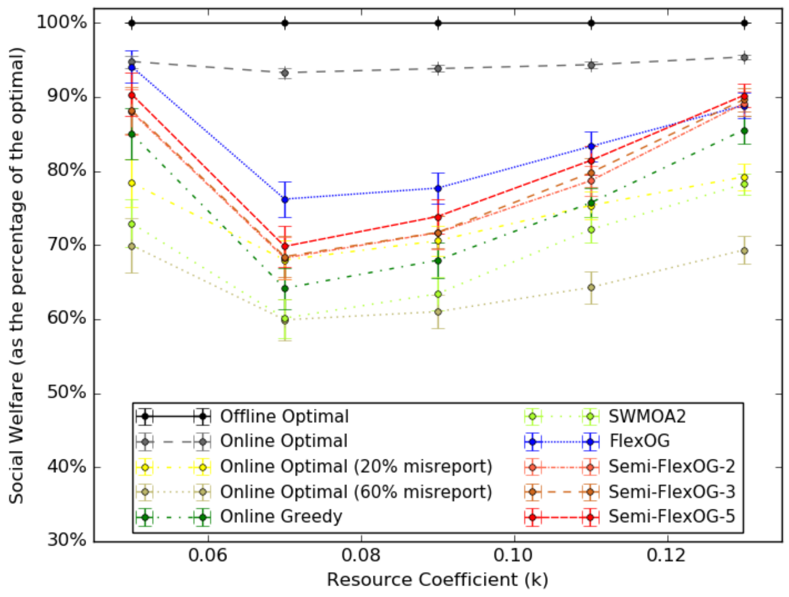

FIGURE 3.17: Social welfare achieved by five algorithms with all connected network.

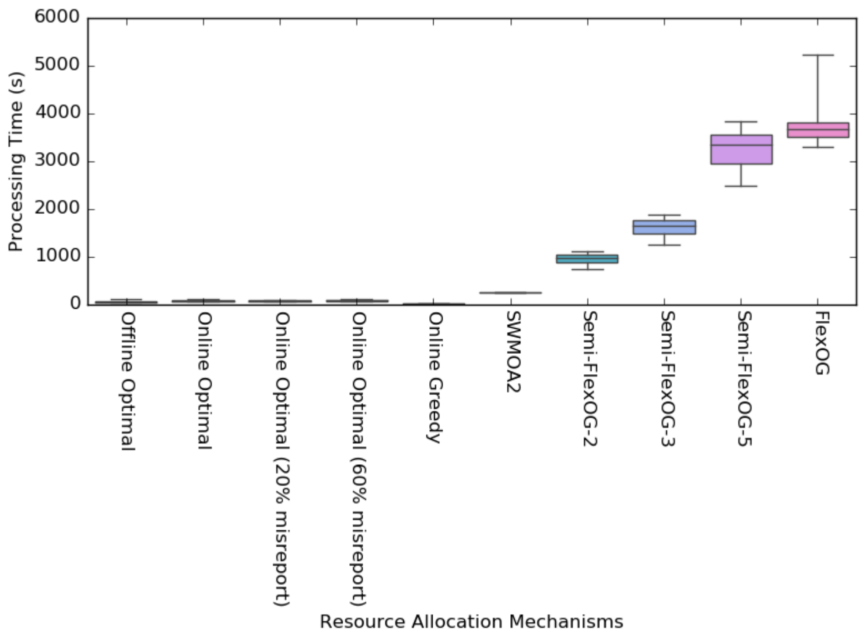

FIGURE 3.18: The processing time by five algorithms with all connected network (k=0.09).

3.4.3.2 Processing Time of Different Algorithms

Figure 3.18 shows the processing time of five resource allocation algorithms. As shown in the figure, Semi-FlexOG takes less processing time than FlexOG on average. Note that the Semi-FlexOG makes a trade-off between scalability and efficiency in terms of social welfare by setting the α parameter. (e.g., when α is smaller, the processing time is

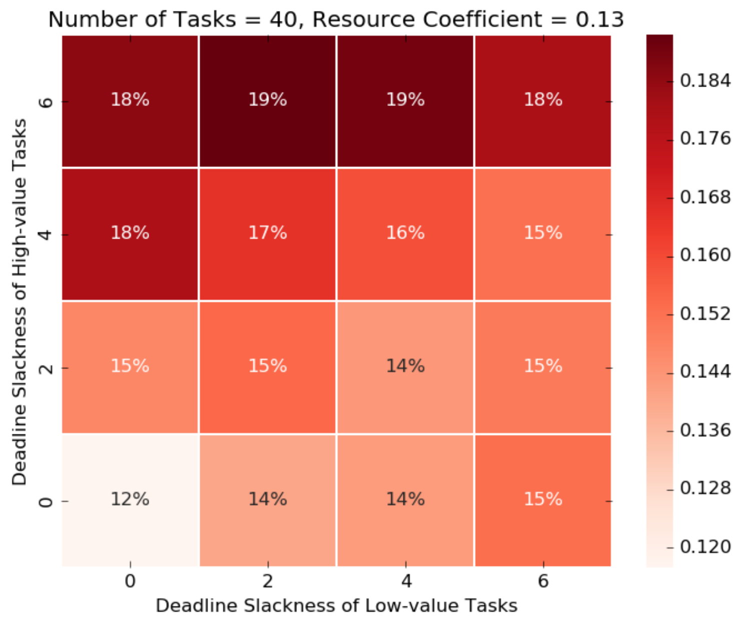

FIGURE 3.19: How much percentage is the social welfare performance of Semi-FlexOG better than that of Online Greedy on average.

low and the social welfare is also lower) In addition, the relative MIP gap tolerance for Offline Optimal is set to be 1%, and the relative MIP gap tolerance in Online Optimal and FlexOG is set to be 5%. (Cplex terminates after a feasible integer solution that is within the relative MIP gap tolerance of optimum is discovered.)

3.4.3.3 Social welfare for Different Levels of Deadline Slackness

Figure 3.19 shows how much percentage Semi-FlexOG outperforms Online Greedy in social welfare (60 trials). As shown in the figure, Semi-FlexOG outperforms Online Greedy when the deadline slackness of high-value/low-value tasks is higher. The start time is generated uniformly between zero and four time slots after the arrival time. The usage time is drawn uniformly between one and three time steps. Furthermore, the level of flexibility of Semi-FlexOG is set to be six here.

3.5 Summary

We detailed the RAFC model studied in this thesis and formalised it as a constraint optimisation problem in this chapter. Then, for the RAFC problem, we introduced a class of resource allocation methods called price-based mechanisms, which are guaranteed to be DSIC and IR, and our proposed mechanism FlexOG belongs to this class of mechanisms. After that, we gave the algorithms of the benchmarks we used to evaluate the performance of FlexOG and presented some properties of these benchmarks. Finally, we described FlexOG in detail and proved that it is DSIC and IR.

In this chapter, we described the synthetic data used in our simulations and analysed the results from a number of empirical simulations. These results highlighted the advantages of the FlexOG mechanism, which consistently outperformed all truthful benchmarks and achieved a social welfare near the theoretical upper bound. Furthermore, we also showed that IoT users are incentivised to misreport their types under the non-truthful benchmark online optimal, which degrades its social welfare significantly. Specifically, online optimal is outperformed by FlexOG when only 20% of IoT users misreport. Besides, the results showed that the advantage of FlexOG is bigger when resources are neither scarce nor sufficient and when tasks have a higher deadline slackness, which is common in practice. Therefore, our proposed mechanism FlexOG addresses the high social welfare resource allocation challenge (Challenge 4). Since FlexOG takes considerable time to make an allocation decision for a newly arrived task, to address the timely resource allocation challenge (Challenge 8), we proposed Semi-FlexOG which is scalable and has a close social welfare performance to FlexOG.

Chapter 4

Decentralised Resource Allocation Mechanisms in Fog Computing

In this chapter, we present our work on designing a decentralised auction algorithm for DRAFC, which addresses our Challenges 4 and 9 in Section 1.3. Note that the agents are assumed non-strategic in this chapter. In Section 4.1, we describe the model of DRAFC both in general and formally. Then, we present the algorithms of benchmark algorithms in Section 4.2 and the algorithms in the decentralised auction in Section 4.3. After that, we show the experimental setup and the results of the simulations in Section 4.4. Finally, Section 4.5 summarises the entire chapter.

4.1 Model of DRAFC

In this section, we describe the DRAFC model used in this thesis in detail because we need a suitable model for decentralised resource allocation. First, we give an overview of the DRAFC model in 4.1.1. After that, a formal description of the model is presented in 3.1.2. Since the DRAFC model is similar to the RAFC model in Chapter 3.1, we mainly describe the differences between DRAFC and RAFC.

4.1.1 Overview of the Model

In our DRAFC model, there are multiple geographically distributed fog nodes. These fog nodes are autonomous agents linked by data links, as shown in Figure 4.1. A key point is that there is no central control to make all resource allocation decisions in this model. Here, fog nodes have computational resources (i.e., CPU, RAM, and storage), and data links have bandwidth resources. We assume the bandwidth of each data link is sufficient because any analytics task is only allocated to one fog node. In

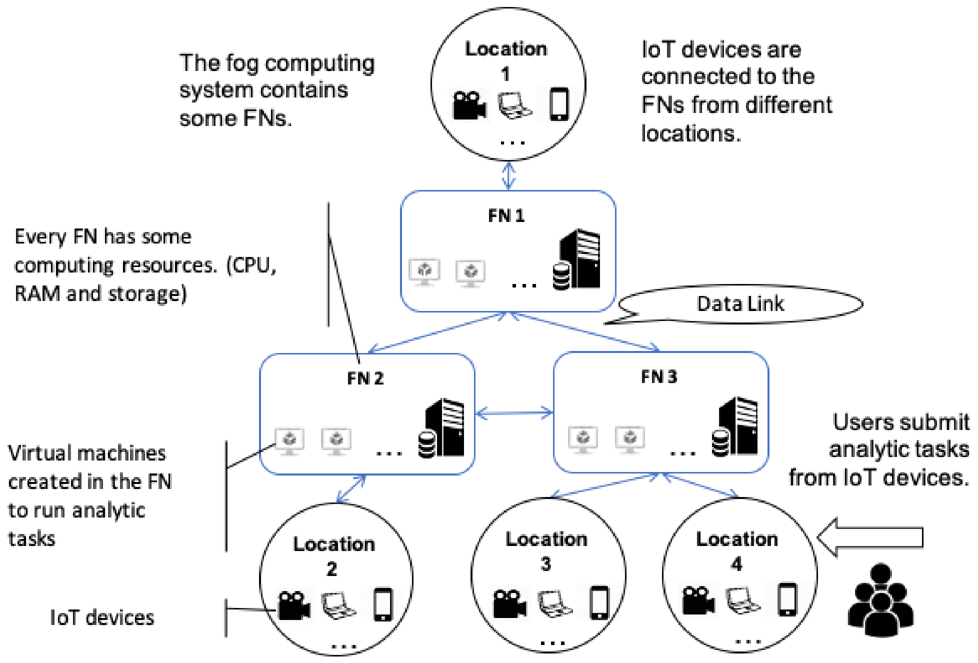
The fog computing system contains some FNs.
Location 1
IoT devices are connected to the FNs from different locations.
Every FN has some computing resources. (CPU, RAM and storage)
FN 1
Data Link
FN 2
FN 3
Users submit analytic tasks from IoT devices.
Virtual machines created in the FN to run analytic tasks
Location 2
Location 3
Location 4
IoT devices

FIGURE 4.1: The general view of the DRAFC model.

other words, no bandwidth is required during processing, and the data link is only used for communications between all fog nodes and uploading necessary data from IoT devices to the fog node. Furthermore, each fog node has its own fixed unit costs of computational resources, which comprise electricity costs, the depreciation charge of the resources and other operational costs. In this model, fog nodes provide VMs to process analytics tasks of the IoT devices, which connect to the fog via wired and wireless connections.

Similar to the RAFC model, IoT users report the type of their analytics tasks to the fog node they connect to. The type includes the following information: resource requirements, time constraints and the valuation function of the task. Similarly, according to the valuation function, a partly processed task can still get parts of its value. In particular, we assume that these tasks are non-preemptive and only one task can only run on one single virtual machine. Furthermore, some users may not get their values if their tasks' results are only released at their deadline. Especially, in contrast to the model in the previous chapter, here we assume fog users are not strategic agents (i.e., they will not misreport their tasks).

However, upon receiving the type of a new task, the fog nodes have to decide the resource allocation plan and the payment of those analytics tasks by communications because there is no central control. Note that in this chapter, we still focus on improving the social welfare (defined in Section 3.1) of all IoT users.

4.1.2 Formal Model of DRAFC

We now formally present the DRAFC model. Since the similarity between the DRAFC and RAFC model, we omit the same assumptions and details. As shown in Figure 4.1 The fog computing system has several fog nodes, given by the set $\mathcal{W}$. We consider a discrete time system, $t = 1, 2, \ldots, T$, where t denotes a discrete time step and T denotes the final time step. Every fog node has three types of resources (i.e., CPU, RAM and storage) that can be used for analytics tasks. The set of all types of resources is denoted by $\mathcal{R}$. fog node $w \in \mathcal{W}$ has a limited quantity of resource $r \in \mathcal{R}$, given by $A_{w,r} \in \mathbb{R}_+$. For example, $A_{1,storage} = 100$ means that fog node 1 has 100G of storage in total. Finally, the unit operational cost of resource $r \in \mathcal{R}$ on fog node $w \in \mathcal{W}$ is denoted as $o_{w,r} \in \mathbb{R}_+$.

Analytics tasks arrive over time and are submitted to different fog nodes. Each task is characterised by a type $\theta_i = \left(\overline{v}_i, t_i, T_i^a, T_i^d, \{a_{i,r}\}_{r \in R}\right)$, where $\overline{v}_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,t_i}]$ is the valuation vector of the task ($v_{i,t}$ denotes the valuation if task i gets t time steps), t_i is the number of time steps for a full completion of task i, T_i^a denotes the arrival time of task i, T_i^d denotes the deadline task i, and $\{a_{i,r}\}_{r \in R}$ is a set denoting the demand for each type of resource of task i per time step. For simplicity, the IoT user submitting task i is also denoted as IoT user $i \in I$ where I is the set of all IoT users. Note that the arrival time of a task means the time that the IoT user realises it needs to process this analytics task, and all tasks are assumed to be non-preemptive (i.e., tasks cannot be paused and resumed again) and can be processed immediately after its arrival or later. Furthermore, the operational cost of task i is denoted as o_i, which is the overall operational cost due to processing task i. Different from the RAFC model, each task can only be processed in one fog node because we believe allocating one task to several fog nodes in a distributed system is too complicated and not practical. In addition, we assume that the reported type θ_i of task i is the same as the true type θ_i, and we leave challenge 5 to future work.

Next, if one fog node receives the type θ_i of task i all the fog nodes will communicate with each other to decide the resource allocation plan λ_i for task i immediately. The plan includes how much usage time $\tilde{t}_i$ will be allocated to task i and the payment p_i of task i, and it will be send to the IoT user i right away. To reduce the communication overhead, once λ_i is decided, it will not change in the future. To show the upper bound of the social welfare of a DRAFC problem, We find the offline optimal social welfare by solving a constraint optimisation problem, with the decision variable: $\{z_{w,t}^i \in \{0,1\}\}_{i \in I, w \in W, t \in T}$, indicating that task i will be processed in fog node w ($z_{w,t}^i = 1$), or not ($z_{w,t}^i = 0$) at time step t. Hence, task i will get usage time $\tilde{t}_i = \sum\limits_{w \in W, t \in T} z_{w,t}^i$ and its resource allocation plan $\lambda_i = \{z_{w,t}^i\}_{i \in I, w \in W, t \in T}$. The objective function of this optimization problem (i.e., Function 4.1) aims to maximise social welfare while adhering to resource and time constraints:

$$\max_{\lambda_i} \quad \sum_{i \in I} v_i\Big(\sum_{w \in W, t \in T} z^i_{w,t}\Big) - \sum_{i \in I, r \in R, w \in W, t \in T} a_{i,r} z^i_{w,t} o_{w,r} \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{w \in W} z^i_{w,t} \leq 1 \ \forall i \in I, t \in T \tag{4.1b}$$

$$\sum_{i \in I} z^i_{w,t} a_{i,r} \leq A_{w,r} \ \forall w \in W, r \in R, t \in T \tag{4.1c}$$

$$z^i_{w,t} = 0 \ \forall i \in I, w \in W, t < \lceil T^a_i \rceil \ or \, t > T^d_i \tag{4.1d}$$

$$\sum_{t=1}^{T-1} |z^i_{w,t+1} - z^i_{w,t}| \leq 2 \ \forall i \in I, w \in W \tag{4.1e}$$

Next, we describe the details of the constraints in the above optimisation problem. First, constraint 4.1b guarantees that each analytics task is only processed in a specific fog node or not processed. Second, constraint 4.1c makes sure that the allocated tasks have enough resources for processing. Third, constraint 4.1d means that any task is only processed after its arrival time and not after its deadline. Finally, constraint 4.1e guarantees that each task is non-preemptive (i.e., task execution does not being interrupted).

However, the fog cannot make resource allocation decisions just by solving this constraint optimisation problem. This is because all information of tasks is required to solve the problem, but the fog must make resource allocation decisions online. Furthermore, this problem is also NP-hard, which means it is intractable. We solve it in this thesis by utilising the IBM ILOG CPLEX Optimization Studio.

Theorem 4.1. *The optimisation problem (4.1) is NP-hard.*

Proof. This proof is the same as the Proof 3.1.2 □

4.2 Resource Allocation Benchmarks

Here, we present the benchmark algorithms used for judging the performance of our decentralised resource allocation algorithm. Specifically, we show the overview of each benchmark along with their pseudocode except for the *offline optimal* algorithm. This is because we can get the performance of *offline optimal* just by solving the constraint optimisation problem 3.1a.

4.2.1 Offline Optimal Algorithm

This algorithm finds the optimal social welfare and resource allocation plan (i.e., the allocation that maximise the overall social welfare) given the types of all future tasks. Note that this benchmark cannot be used in practice because it is a centralised algorithm and needs all tasks' types. In contrast, the DRAFC model requires a decentralised algorithm, and the types of tasks are submitted over time. Hence, this algorithm is only used to show the upper bound of social welfare.

4.2.2 Online Greedy Algorithm

Under this algorithm, a newly arrived task is allocated to the fog node that can maximise its social welfare, and its start time is made as early as possible to make time for later tasks. This benchmark is similar to the Online Greedy (OG) algorithm in Chapter 3, and the main differences are that VM migration is not allowed and tasks are not preemptible. The pseudocode of this algorithm is given in Algorithm 8. Notably, this benchmark is also a centralised algorithm and cannot be directly used in DRAFC.

Algorithm 8: The online greedy algorithm

```
$\Theta_{arrived} \leftarrow \emptyset$                                   // The set of arrived tasks
$\Lambda \leftarrow \{ \}$                        // The dictionary of allocation decisions for each task
for $t$ in $T$ do
    while new tasks arrive within $t$ do
        when a new task $i$ arrives                                    // Tasks arrive over time
        $\Theta_{arrived} \leftarrow \Theta_{arrived} \cup \{i\}$                 // Update the set of arrived tasks
        solve the optimal utility allocation for task $i$ (i.e., $\arg\max_{\lambda_i}(v_i(\lambda_i) - o_i(\lambda_i))$,
          s.t. contraints in Problem 4.1 and given $\Lambda$ & $\theta_i$)      // Find the allocation for
          task $i$ that maximise its social welfare
        choose the $\lambda_i$ whose start time is the smallest if the previous problem has
          multiple solutions.             // Choose the allocation with the earliest start time
        $\Lambda[i] \leftarrow \lambda_i$                                  // Commit this allocation decision
        $p_i \leftarrow o_i(\lambda_i)$          // The payment for task $i$ is its corresponding operational cost
    end while
    allocate resources in time step $(t+1)$ according to $\Lambda$
end for
```

4.2.3 Random Allocation

This is a decentralised algorithm in which a fog node will send the task to a random fog node when it receives it. Then, this random fog node finds the allocation plan for this task that maximise its social welfare (with the earliest start time as well). Therefore, this benchmark is decentralised and can be used DRAFC. Please find the pseudocode of this algorithm in Algorithm 9

Algorithm 9: The random allocation algorithm

```
1  $\Theta_{arrived} \leftarrow \emptyset$                                   // The set of arrived tasks
2  $\Lambda_w \leftarrow \{\ \}, \forall w \in \mathcal{W}$                  // The dictionary of allocation plans for each fog node
3  for $t$ in $T$ do
4      while new tasks arrive within $t$ do
5          when a new task $i$ arrives                               // Tasks arrive over time
6          $\Theta_{arrived} \leftarrow \Theta_{arrived} \cup \{i\}$            // Update the set of arrived tasks
7          send the task to a random fog node $w \in \mathcal{W}$    // Send the task to a random fog
             node
8          solve the optimal utility allocation for task $i$ on fog node $w$ (i.e.,
             $\arg\max_{\lambda_i}(v_i(\lambda_i) - o_i(\lambda_i))$, s.t. contraints in Problem 4.1 , given $\Lambda_w$ & $\theta_i$ & $w$)
             // Find the allocation for task $i$ that maximise its social welfare
9          choose the $\lambda_i$ whose start time is the smallest if the previous problem has
             multiple solutions.             // Choose the allocation with the earliest start time
10         $\Lambda_w[i] \leftarrow \lambda_i$                                    // Commit this allocation plan
11         $p_i \leftarrow o_i(\lambda_i)$              // The payment for task $i$ is its corresponding operational cost
12     end while
13     allocate resources in time step $(t+1)$ according to $\Lambda_w$, $\forall w \in \mathcal{W}$
14 end for
```

4.2.4 Bidding Zero

In this benchmark, a fog node will send the type of the task to all other fog nodes when it receives the request. Next, each fog node finds its own allocation plan that maximises the social welfare of this task (with the earliest start time). After that, they send their bids (i.e., the usage time according to the allocation plan and bidding price = 0) to the fog node that receives the task request. Finally, the task is allocated to the fog node that maximises the social welfare of the task (break ties randomly). The detail of this benchmark is given in Algorithm 10.

4.3 Proposed Resource Allocation Mechanism

In this section, we propose an algorithm that combines MARL and an online reverse auction to maximise the long-term social welfare of the fog (i.e., the gap between the total values of processed tasks and the fog nodes' total operational costs). We adopt a reverse auction to maximise the fog's efficiency in terms of social welfare. Moreover, each fog node uses a reinforcement learning algorithm (i.e., PPO) to learn its bidding policy.

The reverse auction works as follows. Once a IoT device submits a task i to a fog node w^*, that fog node immediately broadcasts the task's type to all other fog nodes in the fog. Then, each fog node will find the maximum number of time steps (i.e.,

Algorithm 10: bidding zero algorithm

```
1  $\Theta_{arrived} \leftarrow \emptyset$    // The set of arrived tasks
2  $\Lambda_w \leftarrow \{\,\}, \forall w \in \mathcal{W}$    // The dictionary of allocation plans for each fog node
3  for $t$ in $T$ do
4      while new tasks arrive within $t$ do
5          when a new task $i$ is submitted to fog node $w^*$    // Tasks arrive over time
6          $\Theta_{arrived} \leftarrow \Theta_{arrived} \cup \{i\}$    // Update the set of arrived tasks
7          broadcast the type of the task to all fog nodes
8          solve the optimal utility allocation for task $i$ on fog node $w$, $\forall w \in \mathcal{W}$ (i.e.,
             $\arg\max_{\lambda_{i,w}}(v_i(\lambda_{i,w}) - o_i(\lambda_{i,w}))$, s.t. contraints in Problem 4.1 , given $\Lambda_w, \theta_i, w$)
             // Each fog node finds the allocation plan for task $i$ that maximise its social
             welfare
9          choose the $\lambda_{i,w}$, $\forall w \in \mathcal{W}$ whose start time is the smallest if the previous
             problem has multiple solutions.    // Choose the allocation plans with the
             earliest start time
10         each fog node $w$ sends its bid $\tilde{b}_{i,w}$ (i.e., $\tilde{t}_{i,w}$ according to $\lambda_{i,w}$ and $\tilde{p}_{i,w} = 0$) to
               fog node $w^*$ // Each fog node sends its bid
11         $w^*$ find the $\tilde{b}_{i,w}$, $\forall w \in \mathcal{W}$ that makes the most social welfare (break ties
               randomly), and sends the task to the fog node $\tilde{w}$ that submitted this bid.
               // Find the bid that makes the most social welfare and send the task fog node $\tilde{w}$
12         $\Lambda_{\tilde{w}}[i] \leftarrow \lambda_{i,\tilde{w}}$    // Commit this allocation decision
13         $p_i \leftarrow 0$    // The payment for task $i$ is 0
14     end while
15     allocate resources in time step $(t+1)$ according to $\Lambda_w$, $\forall w \in \mathcal{W}$
16 end for
```

$\tilde{t}_{i,w}$, $\forall w \in \mathcal{W}$) it can offer to task i and uses an reinforcement learning algorithm to decide on the amount of the bidding price for task i (i.e., the unit price $\tilde{p}_{i,w}$, $\forall w \in \mathcal{W}$ at which it is willing to serve that task) and send the bid $\tilde{b}_{i,w} = \{\tilde{t}_{i,w}, \tilde{p}_{i,w}\}$ to fog node w^*.

Finally, task i is allocated to the fog node that maximises its utility
(i.e., $\tilde{w} = \arg\max_w (g_i - \tilde{p}_{i,w}) \times \tilde{t}_{i,w}$) as long as this utility is positive. The time steps allocated to task i is denoted as $\tilde{t}_i$, and fog node w^* will send task i to fog node $\tilde{w}$ to process. Here, each fog node $w \in \mathcal{W}$ decides its bid $\tilde{b}_{i,w}$ for task i independently without the knowledge of the states of other fog nodes and the types of future tasks. So this is a decentralised online resource allocation algorithm because no single fog node has the complete system information. This algorithm can continue to work even if one or several fog nodes lose their connections.

As for the payment of task i, if the reverse auction is first-price, user i pays $\tilde{p}_{i,\tilde{w}} \times \tilde{t}_{i,w}$ to fog node $\tilde{w}$. If the reverse auction is second-price, user i pays $\tilde{p}_{i,\tilde{w}'} \times \tilde{t}_{i,w}$ to fog node $\tilde{w}$ ($\tilde{p}_{i,\tilde{w}'}$ is the second lowest bidding price).

To make it more clear, Figure 4.2 illustrates the process of this reverse auction, and Algorithm 11 shows its pseudocode.

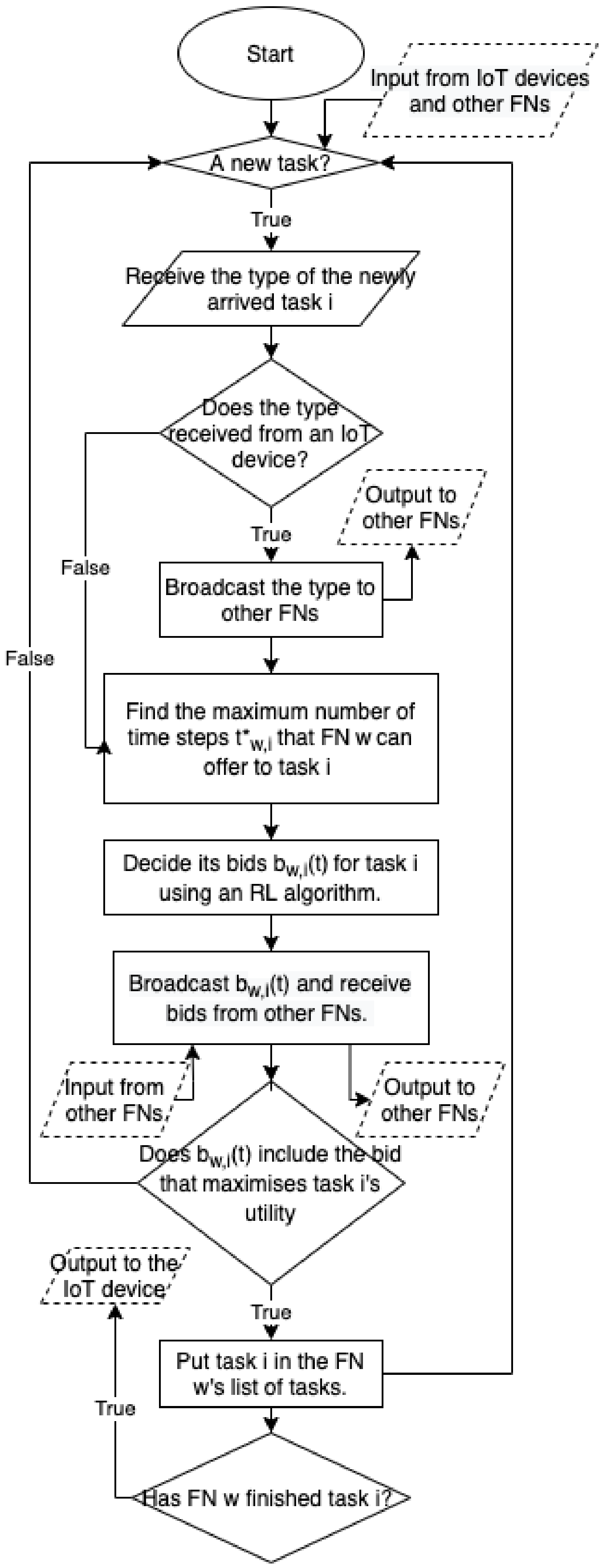
Start
Input from IoT devices and other FNs
A new task?
True
Receive the type of the newly arrived task i
Does the type received from an IoT device?
False
True
Output to other FNs
Broadcast the type to other FNs
False
Find the maximum number of time steps $t^*_{w,i}$ that FN w can offer to task i
Decide its bids $b_{w,i}(t)$ for task i using an RL algorithm.
Broadcast $b_{w,i}(t)$ and receive bids from other FNs.
Input from other FNs
Output to other FNs
Does $b_{w,i}(t)$ include the bid that maximises task i's utility
Output to the IoT device
True
Put task i in the FN w's list of tasks.
True
Has FN w finished task i?

FIGURE 4.2: The flow chart of the reverse auction process in respect of fog node w.

Algorithm 11: DAPPO

```
1  $\Theta_{arrived} \leftarrow \emptyset$    // The set of arrived tasks
2  $\Lambda_w \leftarrow \{\,\}$, $\forall w \in \mathcal{W}$    // The dictionary of allocation plans for each fog node
3  for $t$ in $T$ do
4      while new tasks arrive within $t$ do
5          when a new task $i$ is submitted to fog node $w^*$    // Tasks arrive over time
6          $\Theta_{arrived} \leftarrow \Theta_{arrived} \cup \{i\}$    // Update the set of arrived tasks
7          fog node $w^*$ broadcasts the type of the task to all fog nodes
8          find the maximum usage time for task $i$ on fog node $w$, $\forall w \in \mathcal{W}$ (i.e.,
             $\tilde{t}_{i,w} = \max_{\lambda_{i,w}} (t_{i,w})$), s.t. contraints in Problem 4.1 , given $\Lambda_w, \theta_i, w$)  // Each fog
             node finds the allocation plan for task $i$ that maximise its usage time
9          choose the $\lambda_{i,w}$, $\forall w \in \mathcal{W}$ whose start time is the earliest if the previous
             problem has multiple solutions.    // Choose the allocation plans with the
             earliest start time
10         each fog node $w$ decides its bidding price $\tilde{p}_{i,w}$ using reinforment learning
             algorithm PPO.    // each fog node decides its bidding price
11         each fog node $w$ sends its bid $\tilde{b}_{i,w}$ (i.e., $\tilde{t}_{i,w}$ and $\tilde{p}_{i,w}$) to fog node $w^*$  // Each
             fog node sends out its bid
12         fog node $w^*$ finds the fog node $\tilde{w}$ whose bid $\tilde{b}_{i,w}$, $\forall w \in \mathcal{W}$ makes the most
             utility for task $i$ (i.e., $\tilde{w} = \underset{w}{argmax}(g_i - \tilde{p}_{i,w}) \times \tilde{t}_{i,w}$, break ties randomly),
             and send the task to that fog node.  // Find the fog node that makes the most
             utility
13         each fog node $w$ updates its reinforcement learning model
14         $\Lambda_{\tilde{w}}[i] \leftarrow \lambda_{i,\tilde{w}}$    // Commit this allocation decision
15         $p_i \leftarrow \tilde{p}_{i,\tilde{w}} \times \tilde{t}_{i,\tilde{w}}$    // The payment for task $i$
16     end while
17     allocate resources in time step $(t+1)$ according to $\Lambda_w$, $\forall w \in \mathcal{W}$
18 end for
```

We formulate the DRAFC problem as a Dec-POMDP. In the following, we will describe how the Dec-POMDP, which is defined as a tuple $(W, S, A_w, P(.), R(.), \Omega_i, O, h)$, is formulated.

Agents W: The set of agents W is the set of all fog nodes in the fog.

States S: The state $s \in S$ includes the resource capacity, resource usage of all fog nodes and the resource allocation plans for all arrived tasks. It is important to note that no single node knows the state s of the fog because it only has a partial observation of the system.

Observation o: The observation $o_{i,w}$ of fog node w after the arrival of task i consists of the valuation vector $\overline{v}_i$, the relative deadline (i.e., $T_i^d - \lfloor T_i^a \rfloor$), the resource demands $\{a_{i,r}\}_{r\in\mathcal{R}}$ and the resource utilisation of fog node w in future C time slots (i.e., $\tau_{w,y,t} \in \mathbb{R}$, where $r \in \mathcal{R}$, and $t \in (T_i^a, T_i^a + C]$). Here, $\tau_{w,r,t}$ indicates how much proportion of type r resource in fog node w is occupied in time slot t, which shows how busy is w at time slot t. In summary, $o_{i,w} = \left[\overline{v}_i, (T_i^d - \lfloor T_i^a \rfloor), \{a_{i,r}\}_{r\in\mathcal{R}}, (\tau_{w,r,t})_{r\in\mathcal{R}, t\in(T_i^a, T_i^a+C]}\right]$.

Actions A_w: In order to reduce the number of possible actions, the fog node w has a set of possible actions (e.g., A_w = {0, 0.2, 0.4, 0.6, 0.8}), each element indicating the quotient of its bidding price $\tilde{p}_{i,w}$ and the valuation coefficient g_i of task i. Note that action $a_{i,w} = 0$ means that fog node w rejects task i.

Transition P: The transition function $P(s'|s,a) \in [0,1]$ indicates the probability that state s' happens when action a is taken in state s. Note that a is the joint action of all fog nodes.

Reward Function R: For fog node w, the reward of action a in observation o is the corresponding revenue for fog node w (i.e., $R(w, o_{i,w}, a_{i,w}) = (\tilde{p}_{i,w} - o_{i,w})(\tilde{t}_{i,w})$, where $o_{i,w}$ is the operational cost of task i in fog node w, if the reverse auction is first-price.), if fog node w wins the reverse auction. Otherwise, the reward for fog node w is zero. However, what fog node w tries to maximise in its observation $o_{i,w}$ is the return (i.e., sum of future rewards).

Horizon h: We treat each day as an episode, so h is the end of the day.

Furthermore, since the state-action space of this problem is large, we choose to use neural networks as function approximators to approximate the return of each state-action pair (i.e., $G(o_{i,w}, a_{i,w})$). This can achieve faster computation and more generalisations.

Finally, this problem is a MARL problem. A key challenge of this problem is that the environment faced by each fog node is *non-stationary* because multiple fog nodes learn how to bid concurrently. If the fog node acts like an *independent learner* (i.e., optimises its policy assuming a stationary environment), the reinforcement learning algorithms may fail to converge (Tan, 1993; Claus and Boutilier, 1998). However, independent learning may achieve satisfiable performance empirically (Foerster et al., 2017).

4.4 Simulations and Results

In this section, we assess the DAPPO algorithm for DRAFC through extensive simulations. To begin, we outline the setting in which the synthetic data were generated. Second, we compare the performance of DAPPO with a variety of hyperparameters and in a variety of synthetic data settings. It is shown that DAPPO is more efficient in terms of social welfare than benchmarks and has a close performance to the theoretical upper bound.

4.4.1 Experimental Setup

The setup is similar to that described in Section 3.4.1. We choose the following settings to mimic a modest fog computing network. Our discrete-time period's temporal span ($|T|$) is equal to 12. Thus, each episode corresponds to a single day's allocation of analytics tasks (i.e., one time slot is two hours long). The fog contains six fog nodes ($|P|$=6). Assume that all fog nodes are connected via data links. Additionally, each fog node has $|R| = 3$ distinct types of computational resources (i.e., CPU, RAM, and storage). This arrangement was chosen to represent the following tiny community's fog computing network. This community consists of six dwellings. Each dwelling contains a fog node. In real life, one time step equals two hours, and each day has 24 hours. Another reason we chose this tiny setting is that it allows us to execute more trials for all algorithms in an acceptable amount of time (i.e., our algorithms and benchmarks).

This time period contains $|I| = 40$ tasks. The arrival time Ti^a follows a continuous uniform distribution $U(0, 10)$, which ensures that no two tasks arrive at the same time, as we assumed earlier in our DRAFC model.

Furthermore, for the sake of simplicity, we use a particular valuation function v_i in our simulations, which is a non-decreasing linear function of the real usage time $\tilde{t}_i$:

$$v_i(\tilde{t}_i) = \begin{cases} g_i \times \tilde{t}_i & \text{if } \tilde{t}_i \leq t_i \\ g_i \times t_i & \text{if } \tilde{t}_i > t_i \end{cases}$$

where the valuation coefficient g_i represents task i's obtained value per usage time. This valuation function is the same as the valuation function in Section 3.4.1. Figure 3.5 illustrates an example of this type of valuation functions.

Similar to the experimental setup described in Section 3.4.1, this synthetic data contains two types of tasks: low-value tasks and high-value tasks, and $y \in [0, 1]$ denotes the fraction of high-value tasks. For tasks of low and high value are created using Gaussian distributions $\mathcal{N}(2, 1)$ and $\mathcal{N}(4, 1)$ respectively, with negative values discarded. The time of use ti is a positive number randomly picked from the range $\{1, 2, 3, 4\}$, and all tasks can begin immediately upon arrival. Additionally, the deadline T_i^d is an integer that is uniformly picked between b and d time steps following the earliest finish time (but not exceeding the final time step): $\{T_i^s + t_i - 1 + b, T_i^s + t_i + b, \ldots, min(T_i^s + t_i - 1 + d, |T|)\}$. Thus, (b, d) specifies the task's deadline slackness, a critical variable that shows the task's flexibility. gi is uniformly picked from the continuous interval $[50, 100]$ for a low-value task i . For a high-value task i , on the other hand, g_i is uniformly chosen from the continuous interval $[500, 1000]$. Thus, in this example, the upper constraint on the ratio of the highest and lowest valuation coefficients is $F = \frac{1000}{50} = 20$.

Furthermore, the aggregate resource capacity of each computational resource r is as follows: $\sum_{w \in W} A_{w,r}$ is set to be a k proportion of the overall resource demand: $\sum_{i \in I} a_{i,r}$. Each fog node is then allocated the same percentage of resource r: $\frac{\sum_{w \in W} A_{w,r}}{|W|}$.

Furthermore, we have two types of fog nodes (i.e., low-cost-high-capacity fog nodes and high-cost-low-capacity fog nodes). The resource capacities ($A_{w,r}\, w \in W, r \in R$) of low-capacity fog nodes and high-capacity fog nodes are set to be a k and $\frac{3}{2}k$ fraction of the average resource requests of high-value tasks respectively. Thus, the parameter k reflects the scarcity of the resources in the fog and is called the resource capacity coefficient in this thesis. Moreover, the unit operational cost ($o_{w,r}\, w \in W, r \in R$) of low-cost and high-cost fog nodes is set to be two and four, respectively.

Finally, each fog node has $|\mathcal{A}|$ discrete actions. For example, if each fog node has five actions, $\mathcal{A}$ = {0, 0.2, 0.4, 0.6, 0.8}, where each element indicating the quotient of its bidding price $\tilde{p}_{i,w}$ and the valuation coefficient g_i of task i, and action $a = 0$ means rejecting the task.

4.4.2 Performance of DAPPO

We evaluated the performance of our DAPPO algorithm by running simulations using various parameters, such as the DAPPO algorithm's hyperparameters. Furthermore, we change the resource scarcity in fog nodes, the number of fog nodes in the fog, and the number of tasks in each episode. In particular, DAPPO's performance in social welfare is close to the upper bound (around 90%) and better than benchmarks' performance (0% to 50%). The following sections will show and analyse the outcomes of these simulations.

4.4.2.1 Hyperparameter Optimisation

Hyperparameters can greatly affect the performance of reinforcement learning algorithms. The following results show how different hyperparameters influence the outcome of DAPPO. Note that the y-axis shows the mean social welfare of previous train batch size tasks.

Model Depth Model depth is the number of hidden layers in fully connected neural networks. This experiment investigates the performance of DAPPO with different model depths (each hidden layer has 256 neurones, learning rate = 0.0001, train batch size = 3000, clip param = 0.3), and the results are shown in Figure 4.3[1]. The performance is better for model depth equals one and two, and the performance of DAPPO does not

[1]All figures are within 95% confidence interval based on 100 trials

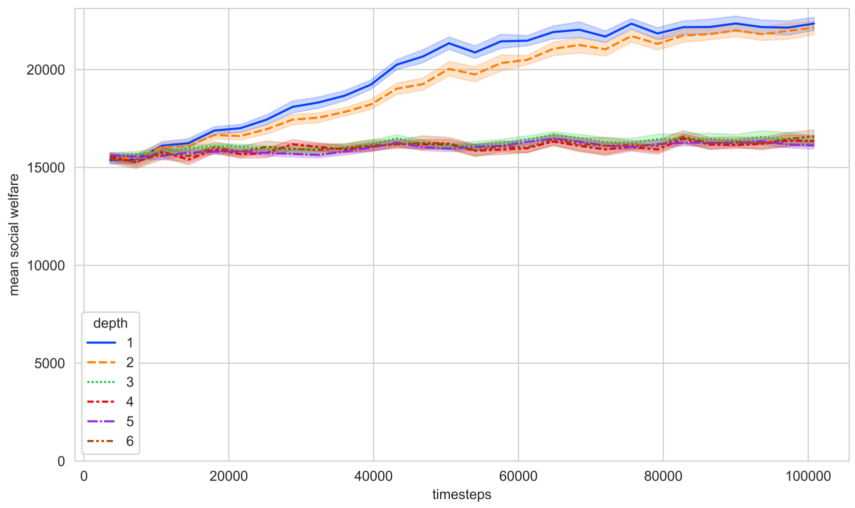

FIGURE 4.3: The mean social welfare achieved by different model depths.

seem to improve during learning for model depth equals three, four, five and six. One explanation could be that deeper neural networks may easily overfit the training data and lead to poor performance.

Learning Rate The learning rate determines the step size in each gradient descend update iteration in PPO. Figure 4.4 illustrates the performance of DAPPO with various learning rates (model depth = 1, train batch size = 3000, clip param = 0.3). They all have similar good performance except for learning rate ≥ 0.01. This is because a large learning rate can make the training unstable. As can be seen from Figure 4.4, DAPPO with a learning rate less or equal to 0.0001 have similar good performance in terms of social welfare.

Train Batch Size Now, train batch size is the number of experiences collected between two gradient descent updates. Typically larger train batch size makes the learning slower but more stable. As shown in Figure 4.5 (model depth = 1, learning rate = 0.0001, clip param = 0.3), DAPPO with smaller train batch size converges faster, and they all achieve similar social welfare at the end.

Clipping Range This hyperparameter, which prevents the updated policy to be too different from the original policy during each policy update, is specific to the PPO algorithm. Figure 4.6 compares the performance of DAPPO with different clipping ranges. From the figure, we can see that DAPPO with clipping range = 0.2 and clipping range = 0.3 have similar good performance, while DAPPO with clipping range 0.1 has significantly lower performance. The most likely cause is that DAPPO with

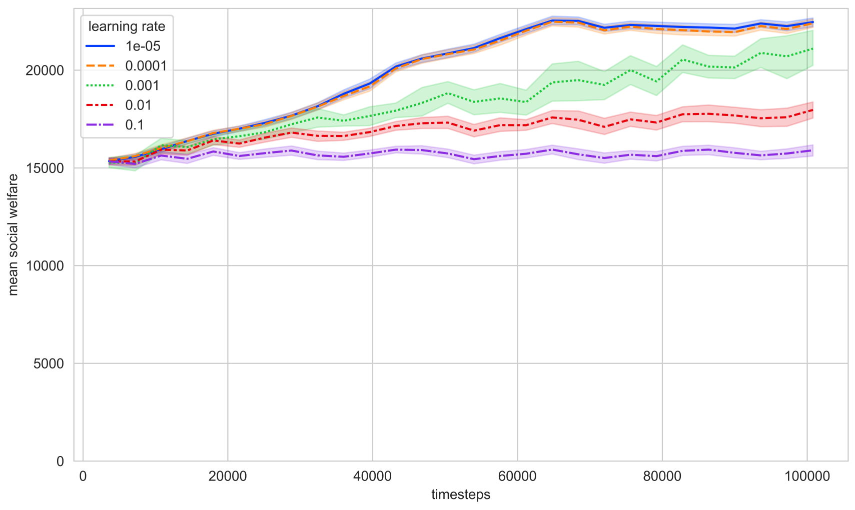

FIGURE 4.4: The mean social welfare achieved by different learning rates.

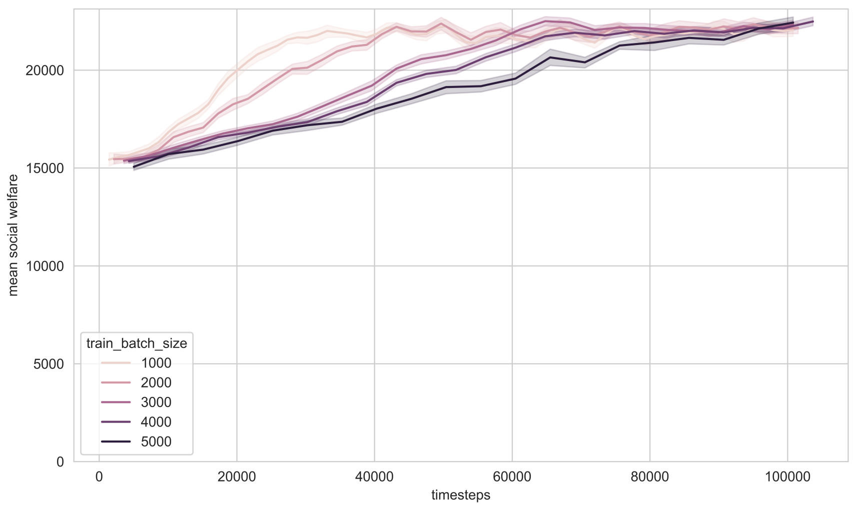

FIGURE 4.5: The mean social welfare achieved by different train batch sizes.

a small clipping range can lead to insufficient exploration and trapped in bad local optima (Wang et al., 2019).

Entropy Coefficient This hyperparameter decides how random the actions of DAPPO, and a bigger entropy coefficient encourages more exploration. It can be seen from Figure 4.7, DAPPO with entropy coefficient 0 or 0.01 have very close performance in terms of social welfare. The most likely cause is that PPO does exploration with both standard deviation and with entropy, and entropy is mainly used to ensure exploration continues during training. So in this setting, the entropy coefficient does not make a

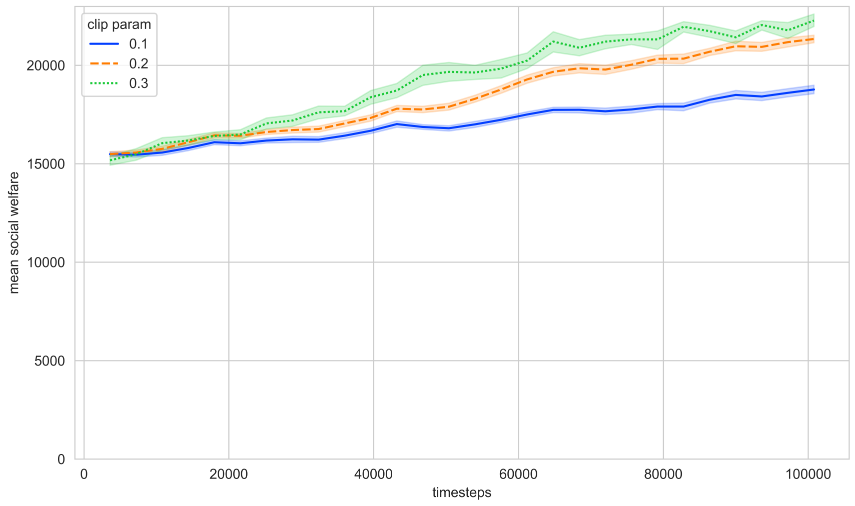

FIGURE 4.6: The mean social welfare achieved by different clipping ranges.

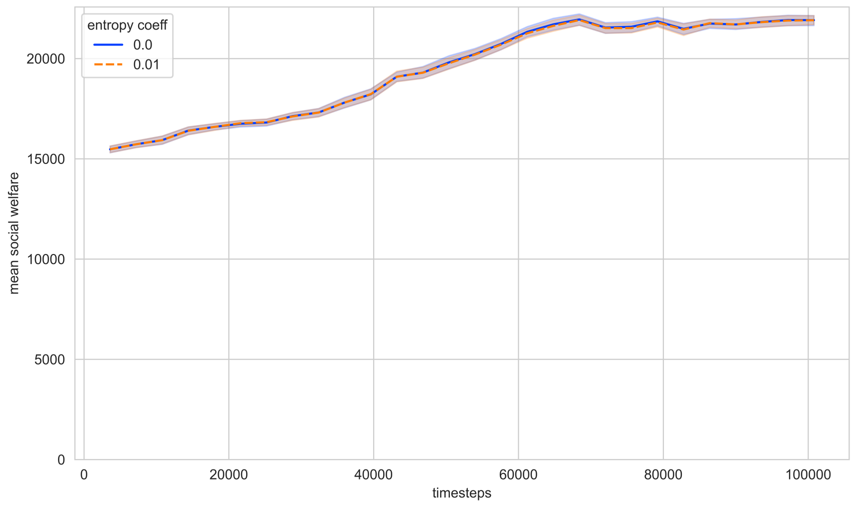

FIGURE 4.7: The mean social welfare achieved by different entropy coefficients.

big difference in the outcome of learning, but if the pattern of tasks changes during training, a higher entropy coefficient could be beneficial.

4.4.2.2 Social Welfare for Different Levels of Resource Scarcity

Next, we compare the social welfare achieved by DAPPO to that of benchmark algorithms when resource coefficient k is varied(model depth = 1, learning rate = 0.001, clip param = 0.3, train batch size = 3000) in Figure 4.8[2]. It can be seen from the figure that DAPPO has a near-optimal performance in terms of social welfare, achieving around 90% of the social welfare of offline optimal for all resource coefficients. This shows that although DAPPO is an online algorithm, it is still quite efficient.

Furthermore, DAPPO has a significantly better performance than other benchmarks except for when the resource coefficient is really big. As shown in Figure 4.8, random allocation always has the worst performance. This is mainly because random allocation may allocate a task to a fog node even if its social welfare is negative. Then, the second-worst benchmark is bidding zero. It has better performance than random allocation because the fog node will reject the task if its social welfare is negative. However, its performance is still not that good because it does not prioritise low-cost fog nodes and is myopic (i.e., it may allocate many low-value tasks and have to reject high-value tasks later). Next, online greedy is a little more efficient than bidding zero because it allocates tasks to low-cost fog nodes first but is still myopic. There are mainly two reasons why DAPPO is more efficient in social welfare than the online benchmarks. The first reason is that the fog nodes will learn that their bidding price should not be lower than the operational cost of the task so that low-cost fog nodes have advantages in the reverse auction. The second reason is that fog nodes can learn to reject tasks with lower values and try to win the bid of the tasks with higher values.

Finally, different resource coefficients k also have a big influence on the performance. In general, the performance of DAPPO is close to online greedy and bidding zero when the resource is either too scarce or too sufficient. First, when the resource is too scarce, the fog nodes can only process low-value tasks because high-value tasks have higher resource requests as well. Second, when the resource is too sufficient, all tasks can be processed. Since online greedy is a centralised algorithm that always prioritises low-cost nodes in resource allocation, it is even a little more efficient than DAPPO when k is big enough.

4.4.2.3 Social Welfare for Different Number of Fog Nodes

Next, we compare the social welfare achieved by different algorithms when the number of fog nodes varies (model depth = 1, learning rate = 0.001, clip param = 0.3, train batch size = 3000, resource coefficient = 1) in Figure 4.9. Note that the number of tasks is proportional to the number of fog nodes (40 tasks for 6 fog nodes), and other parameters are the same as in the previous section. The figure shows that DAPPO always achieves near-optimal social welfare (about 80% to 90%). Furthermore, DAPPO has higher efficiency in terms of social welfare than other benchmarks, and the advantage expands as the number of fog nodes grows. For example, DAPPO's

[2]All figures are within 95% confidence interval based on 100 trials, and the relative tolerance of the CPLEX optimiser is set to be 10% for offline optimal

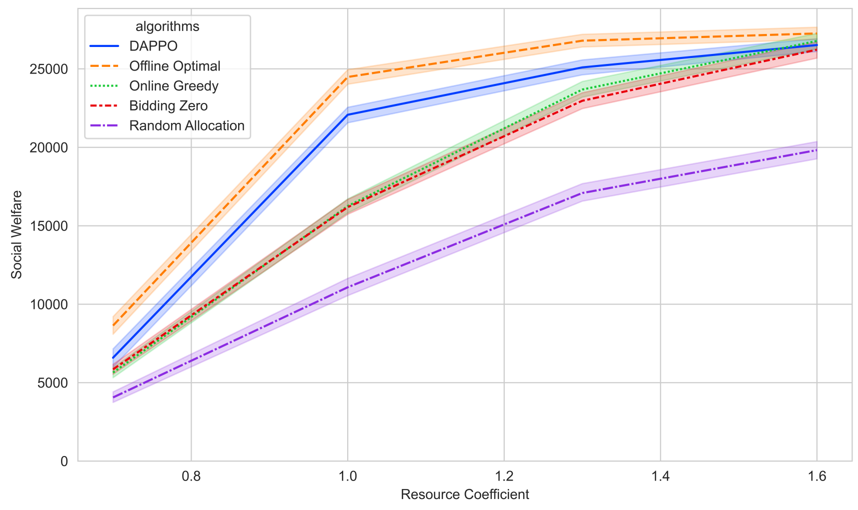

FIGURE 4.8: The social welfare achieved by five algorithms for different resource coefficients.

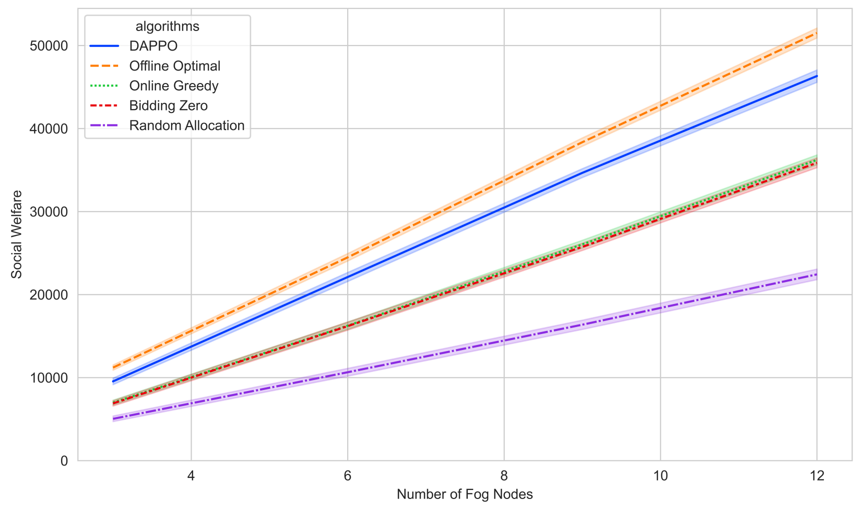

FIGURE 4.9: The social welfare achieved by five algorithms for different numbers of fog nodes.

efficiency is about 20% better than OG when there are two fog nodes, however, DAPPO's efficiency is about 30% better than OG when there are 12 fog nodes. This is mainly because there is more space for optimisation as the number of fog nodes grows and the gap between Offline Optimal and the benchmarks is bigger. Hence, our DAPPO shows a clear advantage in efficiency when there are many fog nodes in the fog.

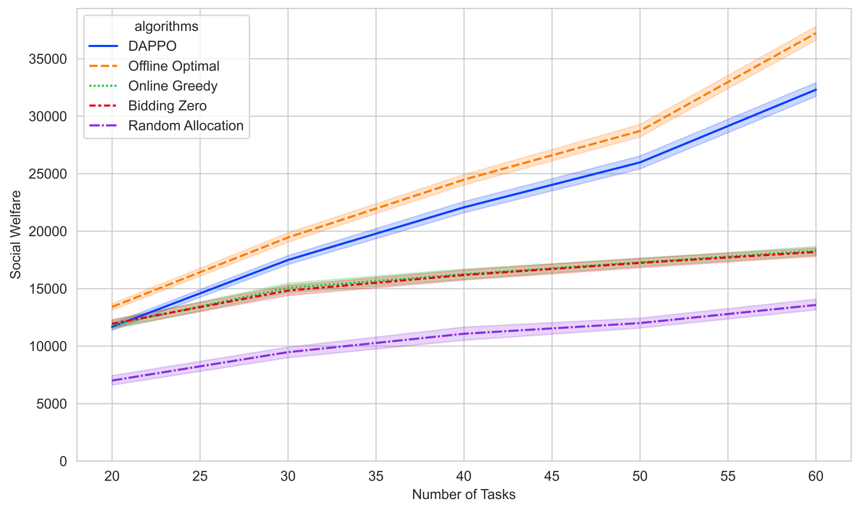

FIGURE 4.10: The social welfare achieved by five algorithms for different numbers of tasks in each episode.

4.4.2.4 Social Welfare for Different Number of Tasks

Now, we compare the social welfare achieved by DAPPO with benchmark algorithms when each episode has a different number of tasks (model depth = 1, learning rate = 0.001, clip param = 0.3, train batch size = 3000) in Figure 4.10. Note that DAPPO is only trained with 40-task episodes. So this experiment shows whether the trained agents are robust to the fluctuations of the number of tasks that arrives in each episode. As can be seen from the figure, DAPPO's efficiency in terms of social welfare is close to (about 80% to 90%) offline optimal and significantly better than other benchmarks except when the number of tasks in each episode is 20. The reason could be that when the number of tasks is small, there is enough resource for both low-value and high-value tasks, and online greedy is already quite efficient in this case. Furthermore, when the number of tasks rises, the performance disparity between DAPPO and benchmarks grows larger. The primary explanation for this is that DAPPO develops a preference for high-value tasks, whereas benchmarks accept an excessive number of low-value tasks and are incapable of accepting high-value tasks that appear later in the episodes. Finally, as shown previously, the performance trends of various algorithms are similar.

4.4.2.5 Social Welfare for Different Types of Reverse Auctions

This section compares social welfare performance for different types of reverse auctions. Note that in the first-price reverse auction, the price is just the bidding price of the winning bid, and in the second-price reverse auction, the price is the same as the lowest bidding price of all losing bids. From Figure 4.11 we can see that DAPPO

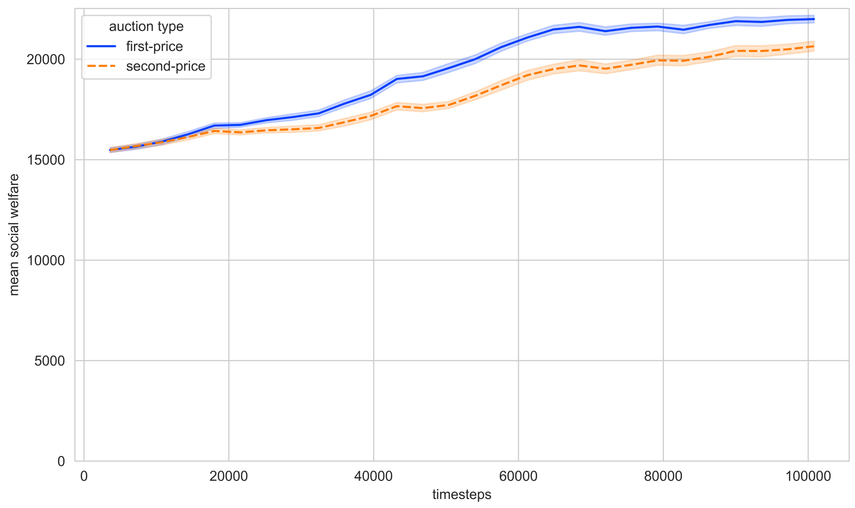

FIGURE 4.11: The mean social welfare achieved by different types of auctions.

with the first-price auction is a little more efficient than DAPPO with the second-price auction. The possible explanation is that in DAPPO with second-price auctions, the reward of the winner is also related to the bidding prices of other fog nodes, which increases the non-stationarity of the environment.

4.4.2.6 Social Welfare for Different Number of Actions

Here, we compare the performance of DAPPO when fog agents have a different number of actions in Figure 4.12. We can see that when each fog node has ten action options, DAPPO converges significantly slower than other cases because more exploration is needed as the number of actions grows. Even so, they all converge to a similar performance after 100000 timesteps.

4.5 Summary

In this chapter, we described the details of the DRAFC model used in the decentralised auction. Then, we listed the algorithms of the benchmarks we used to evaluate the performance of DAPPO. After that, we described DAPPO in detail and the synthetic data used in our simulations. Finally, we presented the learning outcome of DAPPO with different hyperparameters and compared DAPPO's efficiency with the benchmarks in terms of social welfare. The simulation results show that after training, DAPPO can achieve near-optimal social welfare, which is significantly better than that of benchmarks. In particular, DAPPO is also shown to be robust to the fluctuation

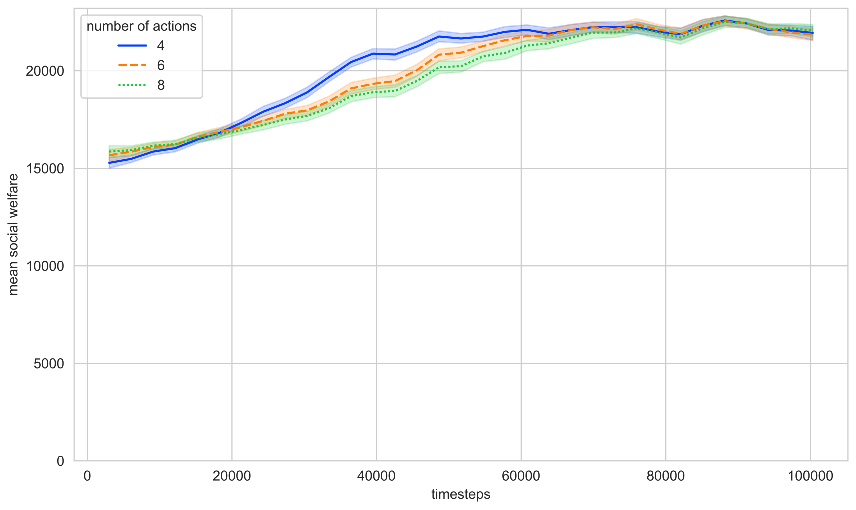

FIGURE 4.12: The social welfare achieved by different number of actions of each fog node.

of task numbers in each episode. Therefore, our proposed algorithm DAPPO can address the high social welfare resource allocation challenge (Challeng 4). Since it can work without a central control, it also addresses the decentralised resource allocation challenge (Challenge 9).

Chapter 5

Conclusions and Future Work

The final chapter summarises this thesis's contributions to the fields of RAFC and DRAFC and suggests future directions for research. To this purpose, in Section 5.1, we first discuss the rationale for this thesis and then present an outline of the techniques we propose to address it. Then, we outline our research contributions and connect them to the research difficulties discussed in Section 1.3, and we summarise the evaluation outcomes of our proposed mechanism by comparing it to benchmark mechanisms. Finally, we discuss probable future works in Section 5.2.

5.1 Research Summary

Nowadays, the IoT is developing very fast and is predicted to revolutionise the way we work and live by making all kinds of physical devices into IoT devices and connecting them to the Internet. In this context, fog computing is proposed to complement cloud computing in providing computing resources to IoT devices. Fog computing has many important benefits, such as low latency and reduced data traffic, which is critical to the popularisation of the IoT. Hitherto there is no truthful resource allocation mechanism that achieves high social welfare in our RAFC model. As we have argued in this thesis, it is reasonable to treat IoT users as intelligent agents who are rational and have their own goals (i.e., utility maximisation). Therefore, the first goal is to design a strategy-proof resource allocation mechanism and thus can be used in a strategic setting. Our second goal is to design a decentralised resource allocation mechanism to avoid a single point of failure and make the whole system more reliable. Note that the fog users are assumed to be truthful in our second goal, so the decentralised allocation mechanism is not required to be strategy-proof.

Reviewing the existing resource allocation mechanisms, we found that they cannot directly apply to our RAFC model and only address a subset of our research challenges.

A lot of existing approaches only focus exclusively on homogeneous resource allocation, whereas RAFC contains multiple heterogeneous resources. Furthermore, most work looks at different types of tasks, which only have value for agents if they are fully accomplished. In addition, most existing mechanisms are not truthful and, therefore, infeasible to work in a strategic setting. Thus, in this thesis (Chapter 3), we proposed a truthful resource allocation mechanism for our RAFC problem. We achieved the first goal by extending a class of mechanisms called price-based mechanisms and using a technique called pre-commitment.

Furthermore, although many reinforcement learning-based approaches are proposed for RAFC, they are not suitable for the DRAFC problem. Specifically, most of these approaches are centralised resource allocation algorithms, and others are letting the IoT devices make resource allocation decisions (i.e., offloading tasks to the fog or not). Hence, in this thesis (Chapter 4) we presented a decentralise resource allocation algorithm for our DRAFC problem that is efficient in terms of social welfare. We achieved the second goal by combining reinforcement learning and online reverse auction.

First, we described the RAFC problem we studied in detail and formulated it as a constraint optimisation problem (Optimisation Problem 3.1a) considering all related research challenges (Challenges 1, 2, 3, 4 and 7). This optimisation problem includes the bandwidth and traffic routing constraints (Challenge 1), and it allows dynamic generation of VMs in any fog nodes (Challenge 2) and uses time-oriented valuation functions (Challenge 3). Furthermore, its objective function includes the operational cost of the fog (Challenge 7) and is to maximise social welfare (Challenge 4). Similarly, we described our DRAFC problem in detail and formulated it as a constraint optimisation problem (Optimisatioin Problem 4.1), which is comprised of Research Challenges 2, 3, 4 and 7. This contraint optimisation problem is similar to the constraint optimisation problem for RAFC and the main difference is that in the optimisation problem for DRAFC each task is only processed in one VM, and there are no bandwidth constraints.

Second, in order to allocate resources efficiently in a strategic setting (Challenges 4, 5 and 6), we designed a price-based online mechanism called FlexOG. This mechanism can be categorised as a price-based mechanism, which ensures that it is truthful and works in a strategic setting. Furthermore, FlexOG uses a technique called pre-commitment, which means that FlexOG only commits the usage time to a task when it arrives but leaves how the usage time is scheduled flexibly. This technique gives FlexOG an advantage in achieving higher social welfare than other truthful benchmark mechanisms. However, it is still possible to further increase the efficiency of our mechanism, so our Challenge 4 is partly addressed. The future work section (Section 5.2) discusses in detail how we intend to boost its efficiency. Furthermore, to increase the scalability of FlexOG, we proposed Semi-FlexOG, which is scalable at the cost of a minor decrease in social welfare. In addition, to allocate resource efficiently in a decentralised setting (Challenges 4, 3, 6 and 9), we proposed a reinforcement learning based online reverse auction called DAPPO. In this reverse auction, each fog node learns the policy that maximises its own revenue using a reinforcement learning algorithm called PPO, which makes it more adaptable in a dynamic environment and no central control is needed, which solves Challenge 9.

Finally, to make sure that FlexOG, Semi-FlexOG, and DAPPO can truly achieve higher social welfare than other state-of-the-art mechanisms for RAFC and DRAFC respectively (Challenge 4), we carried out extensive simulations to evaluate their performance. More specifically, we evaluated FlexOG using synthetic data of different network topologies, resource scarcities or deadline slackness of tasks. Across all of these settings, FlexOG achieves social welfare higher than the two truthful benchmarks (around 5-10% better than OG and 10-20% better than SWMOA2) and close to the theoretical upper bound (around 90%). In particular, we showed through simulations that agents are incentivised to misreport under the non-truthful benchmark online optimal, and the social welfare achieved by online optimal falls below FlexOG when just 20% of agents misreport their valuation functions. In short, these simulations indicated that FlexOG is more efficient than all benchmark mechanisms in a strategic setting. However, the simulations show that the processing time of FlexOG is the highest, and it takes about $2 - 6$ seconds to make a decision for a newly reported task. So we evaluated Semi-FlexOG and showed that its processing time is significantly lower than FlexOG. Although the efficiency of Semi-FlexOG is lower than FlexOG, Semi-FlexOG still outperforms two truthful benchmarks(i.e., OG and SWMOA2). For the DRAFC problem, we evaluated DAPPO using synthetic data of different resource scarcities and a various number of tasks in each episode. Additionally, we also compared the performance of DAPPO with different hyperparameters and found a good combination of hyperparameters for DAPPO. In detail, for different resource scarcities, DAPPO has better efficiency in terms of social welfare than benchmarks (around 0-30% better than OG and Bidding Zero, and 10-50% better than Random Allocation) and close to the upper bound (around 90%). In particular, DAPPO trained by 40 tasks in each episode consistently outperforms benchmarks (around 0-100%) and is close to the upper bound (around 90%) when the number of tasks in each episode changes (from 20 to 60 tasks in each episode).

5.2 Future Work

This thesis' work can be extended in a variety of ways. First, to make their performance more convincing, an empirical evaluation of resource allocation mechanisms with real data evaluation or using fog computing simulation tools could be future work, and we detail these in Section 5.2.1. Second, we could try to further increase the scalability of FlexOG. Another future work could be designing a truthful and decentralised mechanism for the DRAFC problem so that Challenge 9 and 5 can be addressed simultaneously. We discuss how this might be done in Section 5.2.2.

5.2.1 Empirical Evaluation

Although our evaluation is performed over a variety of configurations using synthetic data, it is still of value to evaluate our proposed algorithms using real data. Although there is currently a lack of comprehensive real-world data from fog computing systems, one way to circumvent this problem is to combine real-world cloud computing data with synthetic data. For example, we can configure the amount of resource required by a VM based on the resource requirements of individual tasks in Google cluster data or irids5 data[1] and generate other data such as the number of IoT devices, bandwidth demands and required usage time of each task synthetically. Some work takes this approach to evaluate their mechanisms (Zhang et al., 2018c; Shi et al., 2017; Rublein et al., 2021). We can also generate the arrival time of agents from the *fingerprinting* dataset (Uluagac, 2014), which contains the inter-arrival time information from different applications in 30 wireless devices (e.g., iPhones, iPads, Kindles and Netbooks).

5.2.2 Resource Allocation Mechanisms

First, we intend to further enhance our mechanism's scalability to fully address our Challenge 8 because fog providers want to save energy on making resource allocation decisions. Although compared with FlexOG, Semi-FlexOG has decreased the processing time, it is still slower than the benchmarks (e.g., SWMOA2 and online greedy). Furthermore, our simulations are only in a small setting (e.g., six fog nodes, six locations, 12 time steps and 40 tasks). In reality, fog computing systems can have thousands of fog nodes and tasks. Therefore, we may further improve the scalability of our mechanisms. One approach is to simplify the associated data routing problem. For example, we could restrict the number of hops from IoT devices of a task to the VM of that task, which is also in accordance with the low latency requirements of IoT applications. Another option is to design an efficient algorithm to solve the MILP problem of social welfare maximisation instead of solving it using CPLEX. Finally, we are also considering using sub-optimal heuristics, for example, by greedily allocating tasks based on their value densities. Since finding the optimal allocation for newly-arrived tasks is critical to the truthfulness of FlexOG, we would need to ensure that the DSIC and IR properties still hold under the new mechanism, which is challenging.

[1]https://hpc.soton.ac.uk/redmine/projects/iridis-5-support/wiki

Second, we have designed DAPPO, which combines online reverse auctions and reinforcement learning techniques instead of price-based mechanisms to allocate resources. However, under this algorithm, IoT users may get more utility by misreporting their tasks. For example, a task that would be rejected by all fog nodes may get allocated by increasing its valuation coefficient. Fog nodes can also influence what the algorithm learns and gain an advantage in the future. Hence, DAPPO is not a truthful mechanism and may not work in a strategic setting. Therefore, it is interesting to modify DAPPO to ensure that any agents cannot be better off at present or in the future by misreporting their tasks. Although truthful centralised reinforcement learning for online resource allocation has been proposed (Stein et al., 2020), DAPPO uses decentralised reinforcement learning. For this reason, we need to carefully design the reinforcement learning mechanism to make it truthful and efficient at the same time. Finally, we could investigate other approaches to multi-agent reinforcement learning to further improve the efficiency of the mechanism in terms of social welfare. For example, use the networked multi-agent reinforcement learning, where fog nodes are able to exchange information, instead of the fully independent multi-agent reinforcement learning.

References

Mohammad Aazam and Eui-Nam Huh. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 687–694. IEEE, 2015.

Bilal H Abed-Alguni, David J Paul, Stephan K Chalup, and Frans A Henskens. A comparison study of cooperative q-learning algorithms for independent learners. *Int. J. Artif. Intell*, 14(1):71–93, 2016.

Raafat O Aburukba, Mazin AliKarrar, Taha Landolsi, and Khaled El-Fakih. Scheduling internet of things requests to minimize latency in hybrid fog-cloud computing. *Future Generation Computer Systems*, 111:539–551, 2020.

Swati Agarwal, Shashank Yadav, and Arun Kumar Yadav. An efficient architecture and algorithm for resource provisioning in fog computing. *International Journal of Information Engineering and Electronic Business*, 8(1):48–61, 2016.

Anubha Aggarwal, Neetesh Kumar, Deo Prakash Vidyarthi, and Rajkumar Buyya. Fog-integrated cloud architecture enabled multi-attribute combinatorial reverse auctioning framework. *Simulation Modelling Practice and Theory*, 109:102307, 2021.

PV Ajitha and Ankita Nagra. An overview of artificial intelligence in automobile industry–a case study on tesla cars. *Solid State Technology*, 64(2):503–512, 2021.

Ahmed Al-Ansi, Abdullah M Al-Ansi, Ammar Muthanna, Ibrahim A Elgendy, and Andrey Koucheryavy. Survey on intelligence edge computing in 6g: characteristics, challenges, potential use cases, and market drivers. *Future Internet*, 13(5):118, 2021.

Rana Saeed Al-Maroof, Aseel M Alfaisal, and Said A Salloum. Google glass adoption in the educational environment: A case study in the gulf area. *Education and Information Technologies*, 26(3):2477–2500, 2021.

May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013.

Ahmad Salah AlAhmad, Hasan Kahtan, Yehia Ibrahim Alzoubi, Omar Ali, and Ashraf Jaradat. Mobile cloud computing models security issues: A systematic review. *Journal of Network and Computer Applications*, 190:103152, 2021.

Abdulrahman Alamer. Security and privacy-awareness in a software-defined fog computing network for the internet of things. *Optical Switching and Networking*, 41: 100616, 2021.

Muhammad Ali, Ashiq Anjum, M Usman Yaseen, A Reza Zamani, Daniel Balouek-Thomert, Omer Rana, and Manish Parashar. Edge enhanced deep learning system for large-scale video stream analytics. In *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–10. IEEE, 2018.

Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, 2017.

Palo Alto. Saguna and gridraster partner to bring bigh-quality vr/ar experiences to mobile devices by leveraging multi-access edge computing. https://www.prnewswire.com/news-releases/saguna-and-gridraster-partner-to-bring-high-quality-vrar-experiences-to-mobile-devices-by-leveraging-multi-access-edge-computing.html, 2018.

Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.

Hany Atlam, Robert Walters, and Gary Wills. Fog computing and the internet of things: a review. *Big Data and Cognitive Computing*, 2(2), 2018.

Nitin Auluck, Akramul Azim, and Kaneez Fizza. Improving the schedulability of real-time tasks using fog computing. *IEEE Transactions on Services Computing*, 2019.

Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 85–100. Society for Industrial and Applied Mathematics, 2013.

Yossi Azar, Inna Kalp-Shaltiel, Brendan Lucier, Ishai Menache, Joseph Seffi Naor, and Jonathan Yaniv. Truthful online scheduling with commitments. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 715–732. ACM, 2015.

Moshe Babaioff, Liad Blumrosen, and Aaron Roth. Auctions with online supply. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 13–22, 2010.

Moshe Babaioff, Ronny Lempel, Brendan Lucier, Ishai Menache, Aleksandrs Slivkins, and Sam Chiu-wai Wong. Truthful online scheduling of cloud workloads under uncertainty. In *Proceedings of the ACM Web Conference 2022*, pages 151–161, 2022.

Ranieri Baraglia, Jose Ignacio Hidalgo, and Raffaele Perego. A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 5(6): 613–622, 2001.

Yair Bartal, Rica Gonen, and Noam Nisan. Incentive compatible multi unit combinatorial auctions. In *Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, pages 72–87. ACM, 2003.

Russell Bent and Pascal Van Hentenryck. The value of consensus in online stochastic scheduling. In *ICAPS*, volume 4, pages 219–226, 2004.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

Fan Bi, Sebastian Stein, Enrico Gerding, Nick Jennings, and Thomas La Porta. A truthful online mechanism for resource allocation in fog computing. In *The 16th Pacific Rim International Conference on Artificial Intelligence*, April 2019a. URL https://eprints.soton.ac.uk/431819/.

Fan Bi, Sebastian Stein, Enrico Gerding, Nick Jennings, and Tom La Porta. A truthful online mechanism for allocating fog computing resources. In *Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, pages 1829–1831, May 2019b. URL https://eprints.soton.ac.uk/430226/.

Luiz F Bittencourt, Javier Diaz-Montes, Rajkumar Buyya, Omer F Rana, and Manish Parashar. Mobility-aware application scheduling in fog computing. *IEEE Cloud Computing*, 4(2):26–35, 2017.

Luiz Fernando Bittencourt, Marcio Moraes Lopes, Ioan Petri, and Omer F Rana. Towards virtual machine migration in fog computing. In *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 1–8. IEEE, 2015.

Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

Piotr Borylo, Artur Lason, Jacek Rzasa, Andrzej Szymanski, and Andrzej Jajszczyk. Energy-aware fog and cloud interplay supported by wide area software defined networking. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2016.

Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700, 2016.

Sylvain Bouveret and Jérôme Lang. A general elicitation-free protocol for allocating indivisible goods. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

Frederick P Brooks. What's real about virtual reality? *IEEE Computer graphics and applications*, 19(6):16–27, 1999.

Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

Bin Cao, Zhiheng Sun, Jintong Zhang, and Yu Gu. Resource allocation in 5g iov architecture based on sdn and fog-cloud computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(6):3832–3840, 2021.

Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1269–1272. ACM, 2011.

Ning Chen, Yu Chen, Yang You, Haibin Ling, Pengpeng Liang, and Roger Zimmermann. Dynamic urban surveillance video stream processing using fog computing. In *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*, pages 105–112. IEEE, 2016.

Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things journal*, 1(4):349–359, 2014.

Xianfu Chen, Honggang Zhang, Celimuge Wu, Shiwen Mao, Yusheng Ji, and Mehdi Bennis. Performance optimization in mobile-edge computing via deep reinforcement learning. *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Aug 2018. .

Yann Chevaleyre, Paul E Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaitre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A Rodriguez-Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30(1), 2006.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

Annalisa Cocchia. Smart and digital city: A systematic literature review. In *Smart city*, pages 13–43. Springer, 2014.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Dominique Demougin and Claude Fluet. Monitoring versus incentives. *European Economic Review*, 45(9):1741–1764, 2001.

Paul Dempsey. The teardown: HTC Vive VR headset. *Engineering & Technology*, 11(7-8): 80–81, 2016.

Shahar Dobzinski and Shaddin Dughmi. On the power of randomization in algorithmic mechanism design. *SIAM Journal on Computing*, 42(6):2287–2304, 2013.

Hao Du, Supeng Leng, Fan Wu, Xiaosha Chen, and Sun Mao. A new vehicular fog computing architecture for cooperative sensing of autonomous driving. *IEEE Access*, 8:10997–11006, 2020.

Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4):1594–1608, 2018.

Lee Alan Dugatkin and Hudson Kern Reeve. *Game theory and animal behavior*. Oxford University Press on Demand, 2000.

Gabriel Evans, Jack Miller, Mariangely Iglesias Pena, Anastacia MacAllister, and Eliot Winer. Evaluating the microsoft hololens through an augmented reality assembly application. In *Degraded environments: sensing, processing, and display 2017*, volume 10197, page 101970V. International Society for Optics and Photonics, 2017.

Muhammad Junaid Farooq and Quanyan Zhu. Adaptive and resilient revenue maximizing dynamic resource allocation and pricing for cloud-enabled iot systems. In *2018 Annual American Control Conference (ACC)*, pages 5292–5297. IEEE, 2018.

Rainer Feldmann, Martin Gairing, Thomas Lücking, Burkhard Monien, and Manuel Rode. Selfish routing in non-cooperative networks: A survey. In *International Symposium on Mathematical Foundations of Computer Science*, pages 21–45. Springer, 2003.

Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future generation computer systems*, 29(1):84–106, 2013.

Tiago M. Fernández-Caramés, Paula Fraga-Lamas, Manuel Suárez-Albela, and Miguel Vilar-Montesinos. A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard. *Sensors*, 18(6), 2018. ISSN 1424-8220. . URL https://www.mdpi.com/1424-8220/18/6/1798.

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1146–1155, 2017.

B Friedlander. A decentralized strategy for resource allocation. *IEEE Transactions on Automatic Control*, 27(1):260–265, 1982.

Eric J Friedman and David C Parkes. Pricing wifi at starbucks: issues in online mechanism design. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 240–241. ACM, 2003.

Jun-Song Fu, Yun Liu, Han-Chieh Chao, Bharat K Bhargava, and Zhen-Jiang Zhang. Secure data storage and searching for industrial iot by integrating fog computing and cloud computing. *IEEE Transactions on Industrial Informatics*, 14(10):4519–4528, 2018.

Guoju Gao, Mingjun Xiao, Jie Wu, He Huang, Shengqi Wang, and Guoliang Chen. Auction-based vm allocation for deadline-sensitive tasks in distributed edge cloud. *IEEE Transactions on Services Computing*, 2019.

Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

Pegah Gazori, Dadmehr Rahbari, and Mohsen Nickray. Saving time and cost on the scheduling of fog-based iot applications using deep reinforcement learning approach. *Future Generation Computer Systems*, 110:1098–1115, 2020.

Enrico H Gerding, Valentin Robu, Sebastian Stein, David C Parkes, Alex Rogers, and Nicholas R Jennings. Online mechanism design for electric vehicle charging. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 811–818. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Enrico H. Gerding, Sebastian Stein, Valentin Robu, Dengji Zhao, and Nicholas R. Jennings. Two-sided online markets for electric vehicle charging. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 989–996, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-1993-5. URL http://dl.acm.org/citation.cfm?id=2484920.2485076.

Alex Gershkov and Benny Moldovanu. Efficient sequential assignment with incomplete information. *Games and Economic Behavior*, 68(1):144–154, 2010.

Peter Gradwell and Julian Padget. Distributed combinatorial resource scheduling. In *Proc. AAMAS Workshop on Smart Grid Technologies (SGT-2005)*, 2005.

Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

Yunan Gu, Zheng Chang, Miao Pan, Lingyang Song, and Zhu Han. Joint radio and computational resource allocation in iot fog computing. *IEEE Transactions on Vehicular Technology*, 67(8):7475–7484, 2018.

Varun Gupta, Benjamin Moseley, Marc Uetz, and Qiaomin Xie. Stochastic online scheduling on unrelated machines. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 228–240. Springer, 2017.

Mohammad T Hajiaghayi. Online auctions with re-usable goods. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 165–174, 2005.

Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C Parkes. Adaptive limited-supply online auctions. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 71–80, 2004.

Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, volume 7, pages 58–65, 2007.

Jason D Hartline and Brendan Lucier. Bayesian algorithmic mechanism design. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 301–310, 2010.

Rondik J Hassan, SR Zeebaree, Siddeeq Y Ameen, Shakir Fattah Kak, MA Sadeeq, Zainab Salih Ageed, AZ Adel, and Azar Abid Salih. State of art survey for iot effects on smart city technology: challenges, opportunities, and solutions. *Asian Journal of Research in Computer Science*, 22:32–48, 2021.

Keiichiro Hayakawa, Enrico H. Gerding, Sebastian Stein, and Takahiro Shiga. Online mechanisms for charging electric vehicles in settings with varying marginal electricity costs. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2610–2616. AAAI Press, 2015. ISBN 978-1-57735-738-4. URL `http://dl.acm.org/citation.cfm?id=2832581.2832614`.

Keiichiro Hayakawa, Enrico H Gerding, Sebastian Stein, and Takahiro Shiga. Price-based online mechanisms for settings with uncertain future procurement costs and multi-unit demand. In *Proceedings of the 17th International Conference on*

Autonomous Agents and MultiAgent Systems, pages 309–317. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Linhai He and Jean Walrand. Pricing and revenue sharing strategies for internet service providers. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pages 205–216. IEEE, 2005.

Dorothea Herreiner and Clemens Puppe. A simple procedure for finding equitable allocations of indivisible goods. *Social Choice and Welfare*, 19(2):415–430, 2002.

Farooq Hoseiny, Sadoon Azizi, Mohammad Shojafar, and Rahim Tafazolli. Joint qos-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system. *ACM Transactions on Internet Technology (TOIT)*, 21(4):1–21, 2021.

Leonid Hurwicz. The design of mechanisms for resource allocation. *The American Economic Review*, 63(2):1–30, 1973.

Kashif Hussain, Mohd Najib Mohd Salleh, Shi Cheng, and Yuhui Shi. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52(4):2191–2233, 2019.

Rituka Jaiswal, Reggie Davidrajuh, and SM Wondimagegnehu. Fog computing for efficient predictive analysis in smart grids. In *Proceedings of the International Conference on Artificial Intelligence and its Applications*, pages 1–6, 2021.

Bushra Jamil, Mohammad Shojafar, Israr Ahmed, Atta Ullah, Kashif Munir, and Humaira Ijaz. A job scheduling algorithm for delay and performance optimization in fog computing. *Concurrency and Computation: Practice and Experience*, 32(7):e5581, 2020.

Tyler A Jost, Bradley Nelson, and Jonathan Rylander. Quantitative analysis of the oculus rift s in controlled movement. *Disability and Rehabilitation: Assistive Technology*, 16(6):632–636, 2021.

Sabihe Kabirzadeh, Dadmehr Rahbari, and Mohsen Nickray. A hyper heuristic algorithm for scheduling of fog networks. In *2017 21st Conference of Open Innovations Association (FRUCT)*, pages 148–155. IEEE, 2017.

Ibrahim Haleem Khan and Mohd Javaid. Role of internet of things (iot) in adoption of industry 4.0. *Journal of Industrial Integration and Management*, page 2150006, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Sebastian Krapohl, Václav Ocelík, and Dawid M Walentek. The instability of globalization: applying evolutionary game theory to global trade cooperation. *Public Choice*, 188(1):31–51, 2021.

Chunlin Li, Jun Liu, Weigang Li, and Youlong Luo. Adaptive priority-based data placement and multi-task scheduling in geo-distributed cloud systems. *Knowledge-Based Systems*, 224:107050, 2021.

Juan Li, Yanmin Zhu, Jiadi Yu, Chengnian Long, Guangtao Xue, and Shiyou Qian. Online auction for iaas clouds: Towards elastic user demands and weighted heterogeneous vms. *IEEE Transactions on Parallel and Distributed Systems*, 29(9): 2075–2089, 2018.

Wei-Yu Lin, Guan-Yu Lin, and Hung-Yu Wei. Dynamic auction mechanism for cloud resource allocation. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 591–592. IEEE Computer Society, 2010.

Dapeng Liu, JC Zuniga, Pierrick Seite, H Chan, and CJ Bernardos. Distributed mobility management: Current practices and gap analysis. Technical report, 2015.

Gaocheng Liu, Shuai Liu, Khan Muhammad, Arun Kumar Sangaiah, and Faiyaz Doctor. Object tracking in vary lighting conditions for fog based intelligent surveillance of public spaces. *IEEE Access*, 6:29283–29296, 2018.

Xiaolan Liu, Zhijin Qin, and Yue Gao. Resource allocation for edge computing in iot networks via reinforcement learning. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–6. IEEE, 2019.

Xiaolan Liu, Jiadong Yu, and Yue Gao. Multi-agent reinforcement learning for resource allocation in iot networks with edge computing. *arXiv preprint arXiv:2004.02315*, 2020.

Xilong Liu and Nirwan Ansari. Toward green iot: Energy solutions and key challenges. *IEEE Communications Magazine*, 57(3):104–110, 2019.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Brendan Lucier, Ishai Menache, Joseph Seffi Naor, and Jonathan Yaniv. Efficient online scheduling for deadline-sensitive jobs. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 305–314. ACM, 2013.

Lingjuan Lyu, Karthik Nandakumar, Ben Rubinstein, Jiong Jin, Justin Bedo, and Marimuthu Palaniswami. Ppfa: Privacy preserving fog-enabled aggregation in smart grid. *IEEE Transactions on Industrial Informatics*, 14(8):3733–3744, 2018. .

Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer, 2018.

Bernard Marr. Machine learning in practice: How does amazon's alexa really work? =https://www.forbes.com/sites/bernardmarr/2018/10/05/how-does-amazons-alexa-really-work, 2018.

Lena Mashayekhy, Mahyar Movahed Nejad, and Daniel Grosu. A ptas mechanism for provisioning and allocation of heterogeneous cloud resources. *IEEE Transactions on Parallel and Distributed Systems*, 26(9):2386–2399, 2015a.

Lena Mashayekhy, Mahyar Movahed Nejad, Daniel Grosu, and Athanasios V Vasilakos. An online mechanism for resource allocation and pricing in clouds. *IEEE transactions on computers*, 65(4):1172–1184, 2015b.

Christian Matt. Fog computing. *Business & Information Systems Engineering*, pages 1–5, 2018.

John M McNamara and Olof Leimar. *Game theory in biology: concepts and frontiers*. Oxford University Press, USA, 2020.

M Yasir Mehmood, Ammar Oad, Muhammad Abrar, Hafiz Mudassir Munir, Syed Faraz Hasan, H Muqeet, and Noorbakhsh Amiri Golilarz. Edge computing for iot-enabled smart grid. *Security and Communication Networks*, 2021, 2021.

Mahammad Shareef Mekala and P Viswanathan. A survey: Smart agriculture iot with cloud computing. In *2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*, pages 1–7. IEEE, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Mxolisi Mtshali, Hlabishi Kobo, Sabelo Dlamini, Matthew Adigun, and Pragasen Mudali. Multi-objective optimization approach for task scheduling in fog computing. In *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, pages 1–6. IEEE, 2019.

Dhritiman Mukherjee, Rik Das, Souvik Majumdar, Sourav Ghosh, Sudeep Thepade, and Abhishek Basu. Energy efficient face recognition in mobile-fog environment. *procedia computer science*, 152:274–281, 2019.

Dhritiman Mukherjee, Sudarshan Nandy, Senthilkumar Mohan, Yasser D Al-Otaibi, and Waleed S Alnumay. Sustainable task scheduling strategy in cloudlets. *Sustainable Computing: Informatics and Systems*, 30:100513, 2021.

Kashif Munir and Lawan A Mohammed. Comparing user authentication techniques for fog computing. In *Advancing Consumer-Centric Fog Computing Architectures*, pages 111–125. IGI Global, 2019.

Iyswarya Narayanan, Aman Kansal, Anand Sivasubramaniam, Bhuvan Urgaonkar, and Sriram Govindan. Towards a leaner geo-distributed cloud infrastructure. In *6th USENIX Workshop on Hot Topics in Cloud Computing*, 2014.

Mahyar Movahed Nejad, Lena Mashayekhy, and Daniel Grosu. Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. *IEEE transactions on parallel and distributed systems*, 26(2):594–603, 2015.

Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 1–12, New York, NY, USA, 2000. ACM. . URL `http://doi.acm.org/10.1145/352871.352872`.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge university press, 2007.

Feyza Yildirim Okay and Suat Ozdemir. A fog computing based smart grid model. In *2016 international symposium on networks, computers and communications (ISNCC)*, pages 1–6. IEEE, 2016.

PG Palafox-Alcantar, DVL Hunt, and CDF Rogers. The complementary use of game theory for the circular economy: A review of waste management decision-making methods in civil engineering. *Waste Management*, 102:598–612, 2020.

David Parkes. *Algorithmic game theory, chapter online mechanisms*. Cambridge University Press, 2007.

David C Parkes and Quang Duong. An ironing-based approach to adaptive online mechanism design in single-valued domains. In *AAAI*, volume 7, pages 94–101, 2007.

David C Parkes and Satinder Singh. An mdp-based approach to online mechanism design. *Advances in neural information processing systems*, 16, 2003.

David C Parkes, Dimah Yanovsky, and Satinder Singh. Approximately efficient online mechanism design. *Advances in neural information processing systems*, 17, 2004.

Xiting Peng, Kaoru Ota, and Mianxiong Dong. Multiattribute-based double auction toward resource allocation in vehicular fog computing. *IEEE Internet of Things Journal*, 7(4):3094–3103, 2020.

Nisha Peter. Fog computing and its real time applications. *International Journal of Emerging Technology and Advanced Engineering*, 5(6):266–269, 2015.

Ryan Porter. Mechanism design for online real-time scheduling. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 61–70. ACM, 2004.

Tao Qin, Wei Chen, and Tie-Yan Liu. Sponsored search auctions: Recent advances and future directions. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4): 60, 2015.

Waleed Rafiq, Abdul Wahid, Munam Ali Shah, and Adnan Akhunzada. Internet traffic flow analysis in fog computing: an experimental case study. In *Recent Trends and Advances in Wireless and IoT-enabled Networks*, pages 83–92. Springer, 2019.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14, 2011.

Vincent Ricquebourg, David Menga, David Durand, Bruno Marhic, Laurent Delahoche, and Christophe Loge. The smart home concept: our immediate future. In *2006 1st IEEE international conference on e-learning in industrial electronics*, pages 23–28. IEEE, 2006.

Emmanouil Rigas, Enrico Gerding, Sebastian Stein, Sarvapali Ramchurn, and Nick Bassiliades. Mechanism design for efficient online and offline allocation of electric vehicles to charging stations. *arXiv preprint arXiv:2007.09715*, 2020.

Vasja Roblek, Maja Meško, and Alojz Krapež. A complex view of industry 4.0. *Sage Open*, 6(2):1–11, 2016.

Valentin Robu, Enrico H Gerding, Sebastian Stein, David C Parkes, Alex Rogers, and Nick R Jennings. An online mechanism for multi-unit demand and its application to plug-in hybrid electric vehicle charging. *Journal of Artificial Intelligence Research*, 48: 175–230, 2013.

B Myerson Roger. Game theory: analysis of conflict. *The President and Fellows of Harvard College, USA*, 1991.

Jörg Rothe. *Economics and computation*, volume 4. Springer, 2015.

Caroline Rublein, Fidan Mehmeti, Mark Towers, Sebastian Stein, and Thomas F La Porta. Online resource allocation in edge computing using distributed bidding approaches. In *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 225–233. IEEE, 2021.

Mohammed Mohammed Sadeeq, Nasiba M Abdulkareem, Subhi RM Zeebaree, Dindar Mikaeel Ahmed, Ahmed Saifullah Sami, and Rizgar R Zebari. Iot and cloud computing issues, challenges and opportunities: A review. *Qubahan Academic Journal*, 1(2):1–7, 2021.

Tuomas Sandholm. Automated mechanism design: A new application area for search algorithms. In *International Conference on Principles and Practice of Constraint Programming*, pages 19–36. Springer, 2003.

Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1):46–59, 2018.

Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, (4):14–23, 2009.

Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Elham Semsar-Kazerooni and Khashayar Khorasani. Multi-agent team cooperation: A game theory approach. *Automatica*, 45(10):2205–2213, 2009.

Jaydip Sen. Security and privacy issues in cloud computing. In *Cloud Technology: Concepts, Methodologies, Tools, and Applications*, pages 1585–1630. IGI Global, 2015.

Weijie Shi, Chuan Wu, and Zongpeng Li. Rsmoa: A revenue and social welfare maximizing online auction for dynamic cloud resource provisioning. In *Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of*, pages 41–50. IEEE, 2014a.

Weijie Shi, Linquan Zhang, Chuan Wu, Zongpeng Li, and Francis Lau. An online auction framework for dynamic resource provisioning in cloud computing. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):71–83, 2014b.

Weijie Shi, Chuan Wu, and Zongpeng Li. A shapley-value mechanism for bandwidth on demand between datacenters. *IEEE Transactions on Cloud Computing*, 6(1):19–32, 2015.

Weijie Shi, Linquan Zhang, Chuan Wu, Zongpeng Li, Francis Lau, Weijie Shi, Linquan Zhang, Chuan Wu, Zongpeng Li, and Francis Lau. An online auction framework for dynamic resource provisioning in cloud computing. *IEEE/ACM Transactions on Networking (TON)*, 24(4):2060–2073, 2016.

Weijie Shi, Chuan Wu, and Zongpeng Li. An online auction mechanism for dynamic virtual cluster provisioning in geo-distributed clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(3):677–688, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, L. Sifre, George van den Driessche, Thore

Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

Jagdeep Singh and Parminder Singh. A sustainable resource allocation techniques for fog computing. *Sustainable Development Through Engineering Innovations*, pages 143–151, 2021.

Shailendra Singh and Abdulsalam Yassine. Iot big data analytics with fog computing for household energy management in smart grids. In *International Conference on Smart Grid and Internet of Things*, pages 13–22. Springer, 2018.

Bam Bahadur Sinha and R Dhanalakshmi. Recent advancements and challenges of internet of things in smart agriculture: A survey. *Future Generation Computer Systems*, 126:169–184, 2022.

Satyajit Sinha. State of iot 2021: Number of connected iot devices growing 9% to 12.3 billion globally, cellular iot now surpassing 2 billion. `https://https://iot-analytics.com/number-connected-iot-devices/`, 2021.

Sebastian Stein, Enrico Gerding, Valentin Robu, and Nicholas R Jennings. A model-based online mechanism with pre-commitment and its application to electric vehicle charging. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 669–676. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Sebastian Stein, Mateusz Ochal, Ioana-Adriana Moisoiu, Enrico Gerding, Raghu Ganti, Ting He, and Tom La Porta. Strategyproof reinforcement learning for online resource allocation. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1296–1304, 2020.

Cristina Stolojescu-Crisan, Calin Crisan, and Bogdan-Petru Butunoi. An iot-based smart home automation system. *Sensors*, 21(11):3784, 2021.

Philipp Ströhle, Enrico H Gerding, Mathijs M de Weerdt, Sebastian Stein, and Valentin Robu. Online mechanism design for scheduling non-preemptive jobs under uncertain supply and demand. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 437–444. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

Kehua Su, Jie Li, and Hongbo Fu. Smart city and the applications. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 1028–1031. IEEE, 2011.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Hassan Takabi, James BD Joshi, and Gail-Joon Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 8(6):24–31, 2010.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

Fan TongKe. Smart agriculture based on cloud computing and IOT. *Journal of Convergence Information Technology*, 8(2), 2013.

Eva Marín Tordera, Xavi Masip-Bruin, Jordi Garcia-Alminana, Admela Jukan, Guang-Jie Ren, Jiafeng Zhu, and Josep Farré. What is a fog node a tutorial on current concepts towards a common definition. *arXiv preprint arXiv:1611.09193*, 2016.

Maria Lorena Tuballa and Michael Lochinvar Abundo. A review of the development of smart grid technologies. *Renewable and Sustainable Energy Reviews*, 59:710–725, 2016.

Shreshth Tuli, Shashikant Ilager, Kotagiri Ramamohanarao, and Rajkumar Buyya. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. *IEEE transactions on mobile computing*, 2020.

A. Selcuk Uluagac. CRAWDAD dataset gatech/fingerprinting (v. 2014-06-09). `https://crawdad.org/gatech/fingerprinting/20140609`, 2014.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

Haiyang Wang, Chenghui Zhang, Ke Li, and Xin Ma. Game theory-based multi-agent capacity optimization for integrated energy systems with compressed air energy storage. *Energy*, 221:119777, 2021.

Jianzong Wang, Yanjun Chen, Daniel Gmach, Changsheng Xie, Jiguang Wan, and Rui Hua. pcloud: an adaptive i/o resource allocation algorithm with revenue consideration over public clouds. In *International Conference on Grid and Pervasive Computing*, pages 16–30. Springer, 2012a.

Juan Wang and Di Li. Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. *Sensors*, 19(5):1023, 2019.

Qian Wang, Kui Ren, and Xiaoqiao Meng. When cloud meets ebay: Towards effective pricing for cloud computing. In *INFOCOM, 2012 Proceedings IEEE*, pages 936–944. IEEE, 2012b.

Shudong Wang, Tianyu Zhao, and Shanchen Pang. Task scheduling algorithm based on improved firework algorithm in fog computing. *IEEE Access*, 8:32385–32394, 2020.

Wanyuan Wang, Yichuan Jiang, and Weiwei Wu. Multiagent-based resource allocation for energy minimization in cloud computing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(2):205–220, 2017.

Wei Wang, Ben Liang, and Baochun Li. Revenue maximization with dynamic auctions in iaas cloud markets. In *2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE, 2013.

Yuhui Wang, Hao He, Xiaoyang Tan, and Yaozhong Gan. Trust region-guided proximal policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Chao Wei, Zubair Md Fadlullah, Nei Kato, and Ivan Stojmenovic. On optimally reducing power loss in micro-grids with power storage devices. *IEEE Journal on Selected Areas in Communications*, 32(7):1361–1370, 2014.

Chuan Wu, Zongpeng Li, Xuanjia Qiu, and Francis Lau. Auction-based p2p vod streaming: Incentives and optimal scheduling. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 8(1S):14, 2012.

Yu Xiao and Chao Zhu. Vehicular fog computing: Vision and challenges. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 6–9. IEEE, 2017.

Jie Xu, Heqiang Wang, and Lixing Chen. Bandwidth allocation for multiple federated learning services in wireless edge networks. *IEEE Transactions on Wireless Communications*, 2021.

Bo Yang, Zhiyong Li, Shilong Jiang, and Keqin Li. Envy-free auction mechanism for vm pricing and allocation in clouds. *Future Generation Computer Systems*, 86:680–693, 2018a.

Xiao Yang, Zhiyong Chen, Kuikui Li, Yaping Sun, Ning Liu, Weiliang Xie, and Yong Zhao. Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff. *IEEE Access*, 6:16665–16677, 2018b.

Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42. ACM, 2015.

Shuai Yu, Xin Wang, and Rami Langar. Computation offloading for mobile edge computing: A deep learning approach. *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct 2017. .

Quan Yuan, Haibo Zhou, Jinglin Li, Zhihan Liu, Fangchun Yang, and Xuemin Sherman Shen. Toward efficient content delivery for automated driving services: An edge computing solution. *IEEE Network*, 32(1):80–86, 2018.

Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers*, 65(12):3702–3712, 2016.

Cheng Zhang, Zhi Liu, Bo Gu, Kyoko Yamori, and Yoshiaki Tanaka. A deep reinforcement learning based approach for cost- and energy-aware multi-flow mobile data offloading. *IEICE Transactions on Communications*, E101.B(7):1625–1634, Jul 2018a. .

Deyu Zhang, Long Tan, Ju Ren, Mohamad Khattar Awad, Shan Zhang, Yaoxue Zhang, and Peng-Jun Wan. Near-optimal and truthful online auction for computation offloading in green edge-computing systems. *IEEE Transactions on Mobile Computing*, 19(4):880–893, 2019a.

Hong Zhang, Hongbo Jiang, Bo Li, Fangming Liu, Athanasios V Vasilakos, and Jiangchuan Liu. A framework for truthful online auctions in cloud computing with heterogeneous user demands. *IEEE Transactions on Computers*, 65(3):805–818, 2016.

Jixian Zhang, Ning Xie, Xuejie Zhang, and Weidong Li. An online auction mechanism for cloud computing resource allocation and pricing based on user evaluation and cost. *Future Generation Computer Systems*, 89:286–299, 2018b.

Jixian Zhang, Xutao Yang, Ning Xie, Xuejie Zhang, Athanasios V Vasilakos, and Weidong Li. An online auction mechanism for time-varying multidimensional resource allocation in clouds. *Future Generation Computer Systems*, 111:27–38, 2020.

Xiaoxi Zhang, Zhiyi Huang, Chuan Wu, Zongpeng Li, and Francis Lau. Online auctions in iaas clouds: Welfare and profit maximization with server costs. In *ACM SIGMETRICS Performance Evaluation Review*, volume 43, pages 3–15. ACM, 2015.

Xiaoxi Zhang, Chuan Wu, Zongpeng Li, and Francis CM Lau. A truthful (1-ϵ)-optimal mechanism for on-demand cloud resource provisioning. *IEEE Transactions on Cloud Computing*, 2018c.

Yutong Zhang, Boya Di, Zijie Zheng, Jinlong Lin, and Lingyang Song. Joint data offloading and resource allocation for multi-cloud heterogeneous mobile edge computing using multi-agent reinforcement learning. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019b.

Zijun Zhang, Zongpeng Li, and Chuan Wu. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):23, 2017.

Ruiting Zhou, Zongpeng Li, Chuan Wu, and Zhiyi Huang. An efficient cloud market mechanism for computing jobs with soft deadlines. *IEEE/ACM Transactions on networking*, 25(2):793–805, 2017.

Liehuang Zhu, Meng Li, Zijian Zhang, Chang Xu, Ruonan Zhang, Xiaojiang Du, and Nadra Guizani. Privacy-preserving authentication and data aggregation for fog-based smart grid. *IEEE Communications Magazine*, 57(6):80–85, 2019. .

Yuefei Zhu, Baochun Li, and Zongpeng Li. Truthful spectrum auction design for secondary networks. In *2012 Proceedings IEEE INFOCOM*, pages 873–881. IEEE, 2012.