Wasps, termites and waspmites: Distinguishing competence from performance in collective construction

Seth Bullock*
School of Electronics and Computer Science,
University of Southampton, UK
sgb@ecs.soton.ac.uk
Tel: +44 (0)2380 595776

Dan Ladley
Department of Economics,
University of Leicester, UK

Michael Kerby
School of Electronics and Computer Science,
University of Southampton, UK

January 11, 2012

Abstract

We introduce a distinction between algorithm performance and algorithm competence and argue that bio-inpsired computing should characterise the former rather than the latter. To exemplify this, we explore and extend a bio-inspired algorithm for collective construction influenced by paper wasp behaviour. Despite being provably general in its competence we demonstrate limitations on the algorithm's performance. We explain these limitations, and extend the algorithm to include pheromone-mediated behaviour typical of termites. The resulting hybrid "waspmite" algorithm shares the generality of the original wasp algorithm, but exhibits improved peroformance and scalability.

Bio-inspireed Computing; Swarm Intelligence; Collective Construction; Stigmergy; Evolution

*Corresponding Author.

1 Introduction

Biology has been a rich source of inspiration for novel computational paradigms. Artificial neural networks, swarm intelligence, artificial immune systems, etc., have all arisen as a result of our growing appreciation of the sophisticated "computational" abilities of living systems.

Part of the attraction of bio-inspired computing is the promise of *general purpose* systems. For instance, a common class of continuous-time recurrent artificial neural network (CTRNN) can be shown to approximate any dynamical system to an arbitrary degree of accuracy [23]; a swarm of artificial insects can implement a general-purpose optimisation algorithm [6]; a cellular automaton is capable of universal computation [17]; wasp and termite behaviour might be idealised to deliver general-purpose construction algorithms for self-organising architectures [7, 27]; and artificial immune systems offer the ability to efficiently classify arbitrary classes of patterns, e.g., [46].

In this paper we challenge this idea, explaining the limited potential for general-purpose performance offered by bio-inspired algorithms, and demonstrating this in the specific case of collective construction algorithms implemented computationally.

2 Competence vs. Performance

The provable generality of some idealised, bio-inspired algorithms suggests that they might enjoy very wide applicability. However, while it is true that some of the real biological systems that inspired these algorithms are more or less specialised than others, none are general purpose in the sense used by

algorithm designers. This is because natural selection is not in the business of fashioning devices that are general purpose. Even brains and immune systems have not evolved to solve novel problems for the organisms that posess them, but only to generlise over the actual historical ones encountered by the ancestors of those organisms [22]. In fact, no biological species, organism, organ, trait or mechanism has ever evolved to serve the function of solving a class of problems that is wider than the finite set of problems actually encountered by its ancestors so far.

For instance, the behavioural mechanisms that termites use to construct their amazing mounds [11] were not evolved for construction, *per se*, but for constructing termite mounds specifically [32, 33]. Our immune system has not evolved to classify arbitrary patterns, but to deal with the particular kinds of pathogen to which we have historically been exposed. Even the human brain, indisputably the most awesome problem-solving mechanism that we know of, is not a general purpose cognitive machine. It is specialised to undertake particular cognitive tasks (language learning, face recognition, social cognition, etc.). It is not organised to solve any problem or deal with every cognitive challenge (witness the large literature on our cognitive shortcomings, e.g., [30]). Rather, the human brain exhibits properties that allow it to successfully tackle the reproductively significant cognitive problems that faced our evolutionary ancestors on the African savannah.

This is not to say that biological mechanisms do not generalise at all. Biological niches are identified with problem classes (digesting a particular *kind* of nut, warning off a particular *species* of predator) rather than just problem instances (digesting a particular nut, warning a particular predator), because

evolution cannot tailor mechanisms to individual problem instances. There must be some exploitable regularity in the instances of a repeatedly encountered problem for them to count as repetitions of the same problem, let alone be solved by a single evolved mechanism. Our immune systems, for instance, have been adapted by evolution such that they are able to cope with many pathogens, some yet to be encountered by our species. We might be tempted to say that our immune system has some predictive ability whenever it copes with a novel pathogen. However, this "predictive ability" is little more than a gamble that the future will resemble the past in particular ways.

Stated more generally, biological devices are shaped by natural selection such that they tend to be well suited to the challenges posed by their Environment of Evolutionary Adaptedness (or EEA, see [22, 9]). This "environment" is actually the sum total of the selection pressures that have been brought to bear on a device's lineage (weighted by recency). It is the finite set of reproductive problems that a particular contemporary biological device's ancestors solved in order that this device (rather than competing forms) currently exists. The EEA is thus similar to the notion of a biological *niche*, in that the character of a biological device can be understood as a reflection (or co-definition) of the demands, pressures, and challenges that characterise its niche. From an alternative perspective, one can expect biological devices to function successfully only under *Normal* conditions: the conditions that the device's ancestors tended to find themselves in historically [36, 37]. Outside such conditions, the performance of an evolved device may be suboptimal, or even pathological (e.g., some forms of human obesity may result from some of our evolved devices operating in a modern

environment featuring many *abNormal* foodstuffs).

This line of argument implies that the biological systems that inspire novel computational architectures, paradigms, or substrates are likely to be well-suited only to *particular* tasks. Even when (idealised abstractions of) these mechanisms are *capable* of exhibiting a very general class of behaviour, we should not expect them to do so *uniformly*—they will tend to be more suited to some tasks than others.

The apotheosis of the claim that a class of system exhibits computational generality is the demonstration of *universality*. Work on universal computation has had profound consequences for our understanding of computing and computability, e.g., [48]. However, demonstrations of general-purposeness, completeness, etc., for CTRNNs, swarm algorithms, genetic encodings, etc., are not typically part of this effort to improve our understanding of computation (some work on cellular automata may be an exception, here). Rather, they are driven by the implicit conviction that the generality of a particular bio-inspired approach is a point in its favour; general schemes, architectures, or algorithms being intrinsically more preferable than specialised ones. If, say, a swarm of artificial ants is demonstrated to simulate a universal Turing machine, the intention is probably to reveal something about the utility of swarm intelligence rather than the nature of computation.

Note that this claim is not as true of the physical systems explored within the fields of unconventional computing or natural computing.[1] Here, there is a stronger tendency to recognise that particular physical systems are suited to solving particular kinds of problem "in materio". However, even within

[1]Thanks to an anonymous reviewer for pointing out this distinction.

these fields there remains an interest in demonstrating that some class of physical system can peform, say, fundamental logical operations and can therefore serve as the substrate for general-purpose computation, e.g., see [14].

The meaning of the word universal derives from parts meaning "all" and "turned towards or against". Hence, whereas universal might be glossed as meaning "all-facing", biological devices are "niche-facing". This will also be true of bio-inspired computational schemes, independent of whether they are provably general or not. For example, while continuous-time recurrent neural networks are capable of exhibiting *arbitrary* dynamics (given enough nodes), it is still true that certain dynamics are *characteristic* of such networks, i.e., this class of device does exhibit a particular *generic* behaviour that can be characterised [3, 12]. Attempting to find or construct networks that exhibit dynamics very different from this generic behaviour is difficult.

Similarly, even if wasp construction behaviours can be idealised such that they are, in theory, capable of generating arbitrary structures [27], it will remain the case that some classes of structure are more readily buildable by such systems. In order to configure such a system to construct architectures that are *uncharacteristic*, one faces a very difficult reverse engineering task that cannot typically be solved by hand and is often even difficult to solve using some kind of powerful search algorithm.

We can use Chomsky's (1965) distinction between *competence* and *performance* to further articulate this issue. While a system's competence corresponds to the range of (linguistic) behaviour that it could produce *in principle*, a system's performance corresponds to the range of (linguistic) be-

haviour that it actually produces *in practice*.[2] Our claim here is that neither knowledge of a bio-inspired algorithm's competence, nor the performance of the system that inspired it are sufficient to support claims of its utility. While we may know that an algorithm's competence is complete, that does not suffice to demonstrate that it will be an efficient or suitable algorithm in all cases. And while the biological systems that inspired it may be impressive, unless we require our bio-inspired system to be merely bio-mimetic (to simply ape the natural activity that inspired it in the first place), we will undoubtedly require its performance to go *beyond* the range of behaviours that are associated with its biological niche, i.e., the performance that we *require* of it will outstrip its *Normal* performance.

A few more words on the competence/performance distinction are required. For Chomsky, linguistic performance is just the actual utterances that *were* in fact made or understood by a language user, while linguistic competence includes the wider set of utterances that the user *could* have used or understood. Here, we want to make the same distinction between the behaviour that a class of algorithm exhibits in practice and the wider range of behaviour that it could exhibit in principle. For instance, consider the behaviour of the perceptron, a class of early neural network [38]. We would use "perceptron competence" to denote the full range of potential behaviour exhibited by the set of all perceptrons under all conditions. By contrast, "perceptron performance" denotes the behaviour of perceptrons that we are likely to encounter in deploying the class of algorithm in the real world. This immediately raises the issue of which parts of the space

[2]Thanks to Richard Watson for suggesting the use of this terminology.

of all perceptrons are likely to be sampled during day-to-day use. It may be that relatively small perceptrons are typically explored, or perceptrons with integer weights. It may be that different communities tend to explore different but perhaps partially-overlapping portions of the space of all perceptrons. It is not necessary to be able to specify the details of these biases and constraints in order to be certain that not every configuration of every perceptron is equally likely to be deployed, and that, as a consequence, perceptron performance will not be equivalent to perceptron competence.

More generally, it is the case that many AI algorithms are parameterised somewhat automatically through a process of search or optimisation. In such situations, a combination of the initial conditions that we tend to choose, any commonalities across the problems that we hope to solve, and the biases of the algorithms that carry out the search or optimisation will determine the generic behaviour of the AI algorithm *in practice*, whatever its range of behaviour *in principle*.

Here we apply this type of thinking to a particular example of a bio-inspired algorithm, demonstrating that although it is inspired by a biological system with impressive performance, and has provably general competence, its own performance is compromised by a combination of inherent limitations and more contingent problems. We use this example to argue for a focus on characterising the generic behaviour of bio-inspired algorithms and how that generic behaviour matches or does not match the nature of the particular problems that we wish to solve.

The practical upshot of characterising algorithm performance rather than competence is to allow practitioners with a particular problem to make

an informed decision as to which algorithm to choose. This point applies across bio-inspired computing in general. However, in the remainder of this paper we explore the idea in the context of a specific class of bio-inspired algorithm.

In the next section, we introduce prior work on collective construction in natural and bio-inspired systems. We then present a specific model inspired by the behaviour of paper wasps [47], and detail its behaviour, before analysing the source of shortcomings in its performance. After a discussion of these and their implications, we explore an extended "waspmite" scheme that incorporates pheromone-mediated behaviour typically employed by species of mound-building termites [32, 33]. We present some readily buildable waspmite structures, before, finally, contrasting the performance and competence of the new scheme with that of schemes that inspired it.

3 Collective Construction

The nests of social insects represent some of the most impressive examples of the extended phenotype seen in nature [18]. In particular, species of termites and paper wasps create structures that demonstrate a high degree of adaption to their environment. Termite mounds exhibit functional heterogeneity with specialised areas set aside for rearing young, or farming "crops". Sophisticated structural organisation allows termites to defend the colony from predators and external changes in temperature. By achieving a form of climate control, fungus gardens can be kept in optimal conditions. Mounds

in different parts of the world exhibit different combinations of these features depending on the local environment [49].

In the case of social wasps, ant predation is believed to be the largest factor influencing the design of the nest [29]. The importance of predation can be seen in various facets of the nest architecture. Nest-building wasp species typically adopt one of two strategies for protecting their nest, and hence their chance of breeding. The first strategy is to cover the nest in a protective shell with a very small opening that is easy to defend in the event of an ant raid. The second strategy is to connect the nest to a tree by a very thin connection, or petiole, coated in a chemical (pheromone) that discourages ants from climbing onto the nest. It is typically not possible for wasps to adopt both strategies simultaneously since multiple petioles are required to support a nest that includes a protective covering—and the task of producing sufficient pheromone to coat each petiole effectively is prohibitive. The exact structure used by a species of wasp is dependent on their environment and evolutionary history, and in particular on the species of ant in their environment [29].

In general, the impressive collective behaviour of group-living insects has been the inspiration for a class of decentralised multi-agent systems known as swarm algorithms [5]. More specifically, studying the construction behaviour of social insects has lead to swarm systems that aim to mimic the sophistication and robustness of their building behaviour: collective construction [52]. Such systems may lead to important insights into construction, with resultant benefits over traditional methods. For instance, a reliance on multiple simple agents gives rise to a degree of fault tolerance that may be absent

from centralised systems. This is especially important when the agents are instantiated as robots in hostile environments such as space [27] or on other planets [28].

Unfortunately, specifying the design of these algorithms is a significant challenge. Predicting the effect of a small change to an agent control system on the observed behaviour of a swarm population has proved very challenging, since system-wide behaviour is typically an emergent property of the interactions of many agents with their environment over time [43].

Rather than attempting to hand-design the control systems for swarm agents, it may be attractive to employ artificial evolution as an automatic design tool. Since natural evolution has led to the design and optimisation of real wasp and termite nests, it would seem reasonable to use artificial evolution to optimise artificial collectively constructed architectures. By creating an appropriate fitness function it should be possible to encourage a computer to do the difficult work of finding agent control algorithms that reliably produce a desired pre-specified structure. Artificial evolution even allows for the possibility of solving design problems by automatically discovering new optimal structures that were not previously considered.

The artificial evolution of structures has been successfully demonstrated in several substrates. Funes and Pollack [24] used a genetic algorithm to artificially evolve LEGO architectures. Constraints on the structure to be formed such as locations that must or must not contain LEGO blocks, and weights of various sizes to be supported at specified locations were translated into a fitness function employed by a genetic algorithm to evolve high-quality structures. Within their model Funes and Pollack evaluated torsional forces

in order to determine if a structure would be stable in the real world, allowing evolved structures to be assembled from real LEGO. This method allowed the design of tables, cranes, bridges and other structures.

Funes and Pollack employ a "direct encoding" in which each element of the structure is explicitly represented along with its relationship to the other structure elements. By contrast, here we are more interested in "indirect encodings" where the locations, relationships and properties of structure elements arise through a generative process of development, the parameters of which are encoded and change over evolutionary time.

One example of such an encoding is that employed by Hornby and Pollack [26] who used "Lindenmeyer" systems [41] to produce table-like structures. During their work they compared evolution using a direct encoding (a list of building instructions) with the evolution of generative production rules that could be used to generate lists of building instructions. They found that the evolution of generative rules was particularly effective as it made good use of modularity within the desired architectures. Again, Hornby and Pollack were able to physically instantiate their simulation results through the use of rapid prototyping techniques.

The formation of termite mound structures has been investigated by Bonabeau et al. [8] and subsequently by two of the authors [32, 33]. In both models the agents were directed solely by local information and had no overall plan of the structure being produced. Rather, the behaviour of the modelled termites was mainly governed by the presence or absence of simulated pheromones. In [32, 33], we demonstrated that artificial termites were capable of building structures reminiscent of those found in termite

mounds using only local information and respecting physical constraints on termite movement and pheromone diffusion.

An earlier example of collective construction was inspired by the behaviour of paper wasps working collectively to build impressive nest structures. Idealising their behaviour resulted in a simple class of decentralised construction scheme capable of generating nest-like structures as well as other interesting architectures [47]. The scheme in some ways resembles the model of termite construction mentioned above in that it involves a swarm of reactive agents moving through a 3-dimensional rectangular lattice, depositing different kinds of building material. However, rather than being driven by the intensity of local pheromone levels, here a wasp agent's building behaviour is determined by a set of production rules, each sensitive to a particular configuration of building material in the 26 cells adjacent to the agent's location (a triggering condition).

With two types of building block available, there are 3^{26} possible triggering conditions. Given that a single set of rules can associate each triggering condition with one of three actions (deposit type-1 block, deposit type-2 block, deposit no block), there are thus $3^{(3^{26})}$ possible rule-sets for building with two block types. Moreover, it is straightforward to demonstrate that, given an arbitrary number of block types, the construction of any configuration of contiguous building material can be specified. In the limit, each block involved in the desired structure may be assigned a unique type, and a rule constructed to specify the unique condition under which it may be placed. This is clearly against the spirit of the scheme, but the same might claimed of the machinations required in order to demonstrate the Turing complete-

ness of, e.g., cellular automata [17]. In practice, since it is likely that the desired structure exhibits some degree of regularity or redundancy, it is expected that the number of unique block types required in order to construct an architecture will be much smaller than the number of blocks deposited. (Determining when the number of block types employed has become so large as to be "too many" is a difficult judgement call. In [27], twenty block types are employed in the construction of 200-block structure.) In any case, in so far as this scheme has this potential to generate an unbounded and exhaustive set of contiguous structures, it can be regarded as general purpose and complete. However, we will demonstrate that the scheme's generality of *competence* does not translate into generality of *performance*.

4 Wasp Collective Construction

In the experiments reported below, the models described by Théraulaz and Bonabeau [47] and Bonabeau et al. [7] are reimplemented. We explore the scheme's ability to generate simple architectures, and develop an account of its performance, i.e., the limits on its behaviour in practice. The intention here is not to assess whether the scheme is a good one or not, but to demonstrate and account for the gap between competence and performance.

4.1 The Basic Model

The simulation was set in a 3d rectangular lattice, $x \times y \times z$, of cubic cells populated by "wasp" agents and blocks of material. The simulation used a fixed discrete time step and was updated synchronously. Each piece of

building material was of a particular type, the number of types used was varied between simulations. "Wasps" could distinguish between blocks of different types. However, there was no other difference between the behaviour of different block types. Agents had two behaviours, movement and block placement, both of which could be performed at every time step. An agent was permitted to move into any of the twenty six adjacent locations in the lattice, the actual destination being chosen at random from a uniform distribution. The movement of the agents was restricted, in that they could not move into locations occupied by building material or move outside the lattice.

Each agent in a simulation had an identical set of rules, termed microrules [47], which governed under which circumstances building material could be placed. Each microrule specified a configuration of the twenty-six neighbouring locations in the world that would stimulate the agent to build. Each location in the microrule could be specified to be either empty or to contain any one of the particular types of building material being used in the simulation. Microrules triggered by a completely empty neighbourhood were not allowed as they quickly lead to space-filling behaviour. In addition microrules were freely rotatable about the vertical axis of the world and so were tested against the local configuration four times in different orientations. If a microrule's stimulating condition was matched, a block of material was placed, the type also being specified by the microrule. Each agent had a set of these microrules, termed an algorithm [47]. When an agent entered a new location, each rule in the algorithm was checked in turn. If a rule was stimulated then the agent deposited material as described. If no rule was

stimulated then no building behaviour occurred. The world was initialised to contain a single piece of building material which would satisfy one of the stimulating conditions of one of the microrules, allowing building to commence. Initially the agents were positioned at random locations within the lattice.

4.2 The Genetic Algorithm

The above model was extended by Bonabeau et al. [7] to include a genetic algorithm, designed to evolve interesting combinations of microrules as defined by a fitness function. In this genetic algorithm the microrule was considered to be the basic unit of evolution. Therefore, each member of the population encoded one combination of microrules. The use of a variable-length encoding allowed the number of microrules in an algorithm to be specified evolutionarily. Each combination of microrules was evaluated once, as described below, and a score awarded by the fitness function.

After each member of the population had been assessed, parents were chosen by roulette-wheel selection, with the probability that a parent algorithm was chosen proportional to its fitness. Two-point crossover was applied during 20% of reproduction events. An offspring algorithm's microrules, inherited during reproduction, were subject to mutation which occurred with probability dependent on whether a microrule had been used in the previous evaluation. If a microrule had been used then there was a very low probability of mutation (0.01). If it had not been used there was a very high probability of mutation (0.9). In the event of mutation a rule was removed from the algorithm and a new rule was generated and inserted in

its place.

New rules were not created at random. Instead, a series of templating functions were used. These templating functions were an attempt to generate rules that were more likely to be used in the simulation than those generated completely at random. The templates specified probabilities for certain microrule configurations. However, they did not entirely prevent any arbitrary configuration from being formed. Additional constraints ensured that any newly placed block had a neighbouring block in at least one cardinal direction and that multiple microrules could not share an identical stimulating condition. These complications were intended to facilitate the evolution of complex non-random structures.[3] Details are provided by Bonabeau et al. [7].

4.3 Fitness Functions

Here we employ three different fitness functions. In each case the starting conditions for a run included a single block at the centre of the ground plane, or, in the first case, the southern vertical face of the lattice.

The first fitness function, F_1, was designed to explore evolving architectures with long range structure, essentially requiring the swarm to place special blocks in locations separated from each other by a set distance. Architectures were rewarded for producing patterns of blocks in a horizontal row beside a starting block placed in the centre of the south face of the lattice. Structures in which a row contains blocks of type 1 that are sepa-

[3]Without these templating functions, our results were essentially the same but evolution took much longer to achieve the same solutions.

rated by at least n blocks of another type or types are favoured. Each such block of type 1 contributes a fitness bonus of +1. For higher values of n, the optimal structure involves long "bridges" of non-type-1 blocks linking individual type-1 blocks. Such structures are necessary whenever distinct structural modules are to be separated from one another by a distance of greater than one cell.

The second fitness function, F_2, was based on the fitness function used by Hornby and Pollack [26] to evolve tables and may be summarised as:

$$F_2 = (f_{height} \times f_{surface} \times f_{stability})/f_{excess}$$

where

$$f_{height} = \text{the height of the highest block, } Y_{max}$$

$$f_{surface} = \text{the number of blocks at } Y_{max}$$

$$f_{stability} = \sum_{y=0}^{Y_{max}-1} f_{area}(y)$$

$$f_{area}(y) = \text{the area of the convex hull at height } y$$

$$f_{excess} = \text{the number of blocks not on the surface}$$

Here the aim is to reward table-like structures with a high top surface that has few gaps in it supported by "legs" between the top surface and the ground. In order to encourage space between the top surface and the ground, the use of blocks that are not part of the surface itself is discouraged.

The third fitness function, F_3, was designed to encourage the wasps to enclose a cubic portion of empty space with four walls. A solid cube of space, measuring $11 \times 11 \times 11$ was defined touching the bottom of the world in the centre. Wasps received a fitness boost for every one of these locations which did *not* contain a block in the finished architecture (+4). In addition, after the building period was complete the presence of walls surrounding the empty space was tested for by firing "rays" from the four sides of the world, travelling horizontally until they hit a block or the opposite side of the lattice. Every time a ray passed through one of the locations in the $11 \times 11 \times 11$ cube, the architecture received a fitness penalty of -1. This function was designed to encourage wasps to place blocks around the central cubic area, but not inside it, i.e., to build an empty cubic structure.

In addition, following Howsman et al. [27], wasps were constrained to be in contact with the surface of the built structure. This resulted in much faster construction as the agents did not spend large amounts of time moving around empty space far away from stimulating conditions. All agents started in contact with the initial block. After an agent placed a block the agent

was moved back into contact with the initial block. In all experiments presented here the initial block was placed next to the border of the world (the locations on the border may not be built in as some of the locations to form the stimulating conditions necessary for building are outside the world) and therefore there was always at least one empty location adjacent to the initial block in which the agents could be placed. Experiments were also performed without this replacement. However, it was observed to have no effect on the fitness of structures discovered.

5 Results

The results obtained for the three fitness functions are described below. The effect of the number of types of blocks available for construction was also investigated for between two and six types of blocks. In all cases a population size of 100 was simulated for 10^4 time steps. First, however, to demonstrate that our reimplentation of the original paper wasp scheme is sound, figure 1 shows a modular repeating structured resembling an insect nest replicated according to the hand-designed micro-rules originally reported by Théraulaz and Bonabeau [47]. The nest is constructed underneath an initial block by first creating a 3×3 horizontal platform two blocks deep immediately below it, then extending this horizontally until a 7×7 block ceiling has been created. A nest wall is built down from the edge of this ceiling and interior floors are constructed at depths of either two or three blocks. The rule-set for constructing this architecture specifies 66 triggering conditions that release building behaviour, and makes use of two types of building block.

[Figure 1 about here.]

5.1 F_1: Bridges

Figure 2 shows typical structures evolved for F_1. Notice that they make use of the different block types to effectively count the required distance between type-1 blocks. For $n \in [1, 2, 3, 4, 5]$ and $T = n + 1$, in the majority of trials the genetic algorithm manages to evolve an optimal rule set, i.e., a rule set which uses the different block types to measure the appropriate distance. However, when n is high some of the trials do not find the optimal structure. They may, for instance, only count four spaces rather than five for $n = 5$.

Figure 3 shows a typical example for $n = 2$ and $T = 2$. Notice that because there are only two block types available, the genetic algorithm isn't able to evolve a simple counting system for measuring distance. Instead a more complex system involving configurations of type-2 blocks is used to measure distance with limited success.

[Figure 2 about here.]

[Figure 3 about here.]

5.2 F_2: Tables

Figure 4 shows a typical structure evolved under F_2. In the vast majority of cases, the genetic algorithm finds this local optimum. A table top is created at ground level. In no cases was the genetic algorithm able to evolve a set of rules that created an elevated table surface. The highest fitness obtained was

not dependent on how many block types the genetic algorithm had access to.

[Figure 4 about here.]

5.3 F_3: Boxes

Figure 5 shows a typical structure evolved under F_3. In all cases, regardless of the number of block types used, the evolved algorithms simply fill space. None are able to produce structures which resemble a box. Again, access to extra block types does not allow the genetic algorithm to evolve a better solution to the problem.

[Figure 5 about here.]

6 Discussion

The results demonstrate the difficulties present in using this system to evolve structures to meet certain criteria. In only one of the cases was the system able to evolve a set of rules that was highly suited to the problem specified by the fitness function, and even then this was only possible under certain conditions ($T \geq n-1$ for F_1). Why did the system perform so poorly? There are two sets of reasons: firstly, inherent limitations of the approach to collective construction; secondly, contingent problems related to the difficulty of exploring the search space.

6.1 Inherent Limitations on Performance

The most crucial weakness of this system is its inability to form non-uniform long-range structures without recourse to an escalating number of block types. By non-uniform long-range structures we mean those structures that cover large spatial distances and do not consist solely of a repeating module, i.e., the rules actively being stimulated in building the structure change over distance.

Modular repetition may occur during a module's construction when a configuration of blocks is produced that is identical to a configuration of blocks that initiated the construction of the module. There is a great deal of potential for this to occur within this system. If any stimulating condition for any previously activated rule, including those triggered by the seed block, is recreated then module repetition may occur. This repetition may be complete, i.e., leading to the creation of a complete new module, or partial, in the sense that a module's construction is interrupted before completion by other repeated parts of the same structure. In order for a structure to be non-repeating, stimulating conditions involved in its construction must not be repeated. This may be extremely challenging, especially with only a small number of block types available.

With a fixed number of block types there are obviously only a finite number of possible stimulating conditions. Initially this would appear to be a vast number, i.e., 3^{26} for $T = 2$ [47]. However, although the total number of stimulating conditions is huge many are not suitable for use, for instance those in which there are no blocks adjacent to the building location in a

cardinal direction. Of those stimulating conditions that are admissible, some occur much more frequently in structures than others, e.g., those consisting of a single block. In particular, once some building activity has occurred, the more common stimulating conditions involving one or two blocks will nearly always be present. If these stimulating conditions are relied upon during the construction of a module, the module has a high potential to repeat. Unfortunately if they are not used in the construction process it is very hard to build a structure as these stimulating conditions are the most common and to some extent the most useful stimulating conditions for building. For instance, a rule which is stimulated by a single block in a horizontally adjacent location could be applied in many places in many structures, but is very useful for building horizontal beams.

This inherent limitation is highlighted in the evolution of bridge structures with two block types (F_1, $T = 2$). In this case the ideal solution is to "count" the desired number of blocks of one type and then place a block of the second type. However, this is not possible with only two block types for $n \neq 1$. When $n = 1$ an alternating pattern of type-1 then type-2 blocks is the optimal solution and this may easily be achieved with two rules. When $n = 2$ the genetic algorithm is able to evolve a more complex rule set which uses pairs of blocks to form unique stimulating conditions. Beyond this point, as n increases the evolution of more complex patterns rapidly becomes more difficult, both in terms of evolving sets of rules which use triples of blocks to identify stimulating conditions, and creating triples which don't re-use stimulating conditions in their own construction. Even in this most simple case where we are only trying to create the most simple

long-range structure, i.e., counting a short distance, the system is not able to perform well. It is easy to imagine tasks more complex than simply measuring a distance, for instance the difficulty of building two connected complex structures. This task would be even more difficult as stimulating conditions could not overlap between the two complex structures or repetition would occur.

The experiments to evolve boxes (F_3) demonstrate this problem. To evolve a high fitness structure it is necessary to evolve a set of rules that can form non-uniform structures over a moderate spatial distance. In the case of boxes the initial block is located in the centre of what would be the base of the box. In order for a box to score highly it is necessary for vertical walls to be formed at a distance of five units from this block. To reliably do this it would be necessary to evolve a method of measuring a set distance and once this distance is reached to start to build vertically. This would require two different building modes and therefore two non-overlapping rule sets.

In order for a rule set to be useful, the structure resulting from it must be able to be constructed reliably. This means that when a given set of rules are used they will result in the formation of the same structure every time they are used. Given that multiple agents with different local perception of the structure are acting simultaneously, there is considerable scope for stochasticity in the building process despite each agent possessing the same deterministic set of building rules. For a structure to be reliably generated from a collective construction algorithm it must be the case that the algorithm is robust to building activity proceeding at multiple sites asynchronously and at random. If this is not the case then a single reliably

constructed structure cannot be guaranteed. In most traditional building application domains, it is likely that this will not be acceptable.

6.2 Contingent Problems for Performance

In addition to problems inherent in this construction methodology there are also problems in evolving and searching for particular types of structures with this system. As was previously shown the system has trouble evolving solutions that have long-range structure. Therefore, the only types of structures that can readily be produced are those that consist of a repeating module and those that are homogeneous, e.g., a plane. Homogeneous structures may be appropriate for some tasks. However, in the majority of tasks they are not good solutions. Also, because planes are homogeneous they suffer from a sparsity of stimulating conditions, i.e., in the extreme case there may be many occurrences of one stimulating condition but no other conditions present. This makes adding useful complexity to a plane difficult.

It is clear from the discussion presented above that structures produced by the system will tend to be modular as a consequence of the decentralised application of a set of local rules. Despite the problems associated with sub-module repetition described previously, modular solutions can be viable. This was demonstrated by Théraulaz and Bonabeau [47], who hand-designed rules to produce some exceedingly complex layered structures based on a relatively large repeating module. Problems, however, start to occur when an attempt is made to evolve modular structures to solve a problem. One of the main difficulties is that often a very poor module that repeats itself will be fitter than a slightly better module that doesn't repeat itself.

Remember that an algorithm will include at least one microrule that is activated by a stimulating condition containing a single block, as this rule was used to start the structure from the seed block. It is very likely that as a module grows through the application of new rules gained through evolution that it will recreate a stimulating condition equivalent to the initial stimulating condition and so lead to repetition. Unfortunately it is very hard to make a module more complex once it has started repeating itself. In order to make a module more complex it is necessary to add a microrule that will be activated during the construction of each module. Unfortunately, adding an extra block to a module will change the stimulating conditions presented by the structure, with the likely consequence of preventing other microrules from being triggered, frequently leading to a drop in fitness.

This problem is very significant. Although at first glance the system may resemble a production rule system, where rules are executed in a set order, this is not the case. The addition of a rule to be applied early in a construction sequence can lead to the loss of all structure after that point. The system is very prone to getting stuck in local optima, as in many cases the addition of any rule would lead to the current structure being lost and so fitness decreasing. Often several rules would need to mutate simultaneously to allow fitness to increase.

The table and box problems demonstrate this tendency to get stuck in local optima. In the case of the table problem, an easy to achieve solution that has poor fitness is to construct a horizontal plane of building blocks at the ground level of the lattice. Unfortunately, once this solution is evolved it is hard to move beyond. If a rule were found that built vertically upwards

from the seed, it would be necessary to simultaneously evolve rules to add blocks to the side of this block before the rules previously used to construct a plane could be applied (the presence of the initial block below the centre means that the rules previously used to construct the plane could not be immediately applied). A similar tendency to discover local optima is exhibited in the box task. It is possible to achieve low fitness by depositing material almost randomly, as some rays will be stopped part way across the world. Even though a box-like structure will score highly in the long run, prior evolutionary stages will score poorly and there is no easy way to transition evolutionarily between the two types of structure.

Therefore, this system will only do well when a poorly designed module that repeats itself scores less highly than a slightly more complex module which doesn't, i.e., the system needs to be forced to avoid local optima. However, to do this often requires prior knowledge of the structure and how it would be created. Where it is available, incorporating such knowledge into the algorithm will necessarily reduce its generality.

7 Possible Solutions

The previous section described difficulties inherent with this system and why evolutionary search will struggle to find algorithms capable of constructing certain classes of structures. However, these results immediately raise concerns as to the adequacy of the genetic algorithm employed here and of the swarm construction scheme itself. Surely it is possible to achieve much better performance by altering one or both of these aspects of the

wasp-agent construction scheme? This section will describe some possible approaches in this respect. However, at the outset we must distinguish between two distinct goals. First we might be interested in simply improving performance on the particular problems presented in this paper. Second, we might be concerned to close the gap between the competence of the scheme and its performance *in general*. Notice that these two aims are often conflated, and that in improving performance on test problems, or benchmarks, or canonical problems, there is often an assumption that improved general performance is automatically being demonstrated or implied.

7.1 More Block Types

Possibly the most obvious solution to the difficulty of forming long-range structure is to use more types of blocks. If larger numbers of block types are used then it is possible to create more unique stimulating conditions. In the limit it is possible to create any contiguous structure given a large enough number of block types. Unique stimulating conditions are necessary as they allow different parts of the construction to differentiate themselves and so allow long-range structure. Without unique stimulating conditions the only options are short range repetition or stimulating conditions which activate more than one rule and so more than one possibility for the structure (though at the cost of some randomness in the final formation).

The use of more block types, however, does not appear to solve the problem. In all but one of the experiments performed with a range of block types available increasing the number of available block types did not significantly increase the fitness of the structures created. One reason for this is the con-

sequent change in the size of the search space. As Théraulaz and Bonabeau [47] showed, given two block types (and 26 neighbouring locations) there are 3^{26} stimulating conditions. Given that each algorithm specifies what a wasp should do in each stimulating condition, i.e., deposit a block or not deposit a block, there are $3^{(3^{26})}$ possible algorithms (again given two block types). If we were attempting to evolve structures of the same complexity as those created by Howsman et al. [27], where twenty distinct types of block are employed in a hand-designed algorithm, the search space of algorithms would be of the order $20^{20^{26}}$ algorithms. As previously noted the search space is highly uncorrelated and so increasing the number of blocks makes the job of finding a reasonable solution to the problem almost impossible.

Only in the most simple cases was the system able to exploit the number of blocks available. This is probably thanks to the templating functions making the rules required for a good solution to this fitness function easier to find, in effect reducing the search space. In most cases the only effect of adding additional block types is to increase the time taken to achieve the same fitness obtained with fewer block types.

7.2 Long-Range Stimuli

A second option, which we explore below, is to give the wasps access to some form of long-range information which they can use to determine which part of the structure they are in and so determine the rules they should follow. One way to do this would be to allow the wasps to use different sets of microrules dependent on their world coordinates. This would allow long-range structures to be created as the wasps would be able to vary

their block placement behaviour dependent on their location. This solution would, however, introduce its own problems. If the world were broken into sectors each with its own rule set, the more sectors there were the more control would be available over long-range structure. However, the search problem would become markedly worse.

In section 8 below we explore an alternative approach inspired by nature. A large amount of termite building behaviour is dictated by the presence of pheromones [11]. Previous work has shown that these pheromones can be used to influence and even dictate the structures produced by agents [8, 32, 33]. A natural extension of this would be to allow every block placed to give off pheromone which could then diffuse in the world. Rules would then have associated with them a range of pheromone values in which they can be activated. If multiple pheromone types were allowed then the architecture could be made to have long-range structure dictated by the pheromone(s) present in a certain area.

There is a strong similarity between this type of pheromone-mediated construction behaviour and biological morphogenesis, where cellular structures arise and organise as a consequence of morphogen-mediated division, differentiation and adhesion. Here, diffusing morphogen chemicals generated by the cells themselves "template" the cellular environment and "program" further developmental activity. In doing so cells can be understood as creating their own co-ordinate system, allowing development to be influenced by self-generated positional information (PI) [56, 57, 42, 44]. These biological phenomena have inspired computational models of artificial development [21, 31, 19], novel amorphus and spatial computation paradigms [39, 1, 25]

and research collective robotics [53].

7.3 Extended Visual Field

A third alternative is to change the visual field associated with the wasps. In reality social insects such as wasps do build long-range structures [29]. One of the ways they are able to do this is through the simultaneous evaluation of multiple cues not limited to the local area [20]. By observing the nest structure the wasps are able to focus their building behaviour in the areas in which it is needed. Instead of restricting the agents' visual field to the 26 neighbouring locations, more remote locations could be sensed [16]. In order to properly integrate some of these ideas it might also be necessary to give each agent a heading. The advantage of this system is that it would allow the wasps to gain long-range information about the structure without necessarily increasing the size of the search space. The disadvantage is that by increasing one aspect of the visual field, i.e., giving the wasp greater visual information from locations in front of it, other aspects of the visual field would have to be sacrificed, in order to avoid increasing the size of the search space. This would mean sacrificing local information in order to gain more long-range information.

7.4 More Sophisticated Control

A fourth alternative is to make the control system for the wasps more sophisticated. Currently the wasps move randomly throughout the world. However, previous work has shown that the way in which agents are allowed to move has a significant effect on the structures created [32]. In addition

to the existing block-placement microrules, movement microrules could also be evolved. This would ensure that, when a wasp encounters a stimulating condition it would be encouraged to make a particular movement. This system would be useful as it would allow wasps to be forced to move to particular places in the world, where if they had an appropriate building rule they would place material. Unfortunately there are two problems with this system. Firstly, evolving movement rules as well as block placement rules would dramatically increase the size of the search space making the job of evolving a rule set significantly harder. Secondly, it could be very difficult to find an appropriate set of movement rules capable of preventing wasps from building in the wrong places at the wrong time, i.e., should a wasp arrive at an inappropriate location.

7.5 More Sophisticated Search

The previous suggestions have all been possible solutions to the inherent problems within this system. However, it is also necessary to solve the contingent problems. The most significant of the contingent problems is the difficulty in using a search algorithm to find suitable rules given a fitness function due to the difficulties in evolving complex modules. One way to solve this problem is to improve the method used to search for rule sets. There are many ways this could be done. For instance, the genetic algorithm could be altered, e.g., through the addition of diversity maintenance techniques that might help to prevent convergence to local optima, or entirely different kinds of search algorithm might be employed.

More radically, the genetic representation of the wasp rule-set could be

revised. Currently, the rule-set is directly specified by genes that each represent an element of a production rule. Consequently there is a one-to-one mapping from genes to parts of rules, and each rule part is free to mutate independently of every other (modulo some of the complications introduced by templating rules, and constraints on duplicating triggering conditions across multiple rules, etc.). An alternative *morphogenetic* scheme would generate a rule set through a process analogous to biological development steered by genes, but not explicitly specified by them.[4] As a consequence, modular rule-sets could become relatively easy to obtain and adapt, since mutations might bring about simultaneous correlated change in a number of developmentally related rules, e.g., [26]. If some kind of environmental stimuli were allowed to influence the developmental processes involved, this might be an effective way of allowing one rule-set to produce different but related structures at different times or in different circumstances. However, developmental encodings of this kind will not be a panacea since allowing some types of structures to be more easily discovered, necessitates that other types will become more difficult or impossible to create. Modification of the search algorithm may improve performance in some cases, but it will not be possible to make a general search algorithm suited to the discovery of any type of structure [54, 55].

[4]Thanks to Jason Noble for suggesting this line of thought.

8 Waspmites

Here we develop and explore a hybrid collective construction algorithm that combines the microrules of the wasp-inspired scheme with pheromone-mediated behaviour typical of termites. We term the resulting agents "waspmites". In doing so, we are adopting a combination of the "Long-Range Stimuli" and "More Sophisticated Control" options described above. There are strong parallels between the resulting scheme and approaches within the study and simulation of morphogeneis, the process by which cellular development generates structure and function in a developing biological organism. We return to these parallels at the end of this seciton. First, we first present the novel scheme in detail, before describing examples of structures built by (hand-designed) waspmites. While the waspmite scheme shares the same competence as the wasp-inspired scheme described above, we demonstrate that it differs significantly in the character of its performance.

8.1 Waspmite Collective Construction

As before, each agent occupies a cell within a 3-d rectangular lattice. At each time-step each waspmite agent is given the opportunity to move to one of the six adjacent locations, excluding those that are occupied by building material of any kind. Movement is also restricted such that waspmites remain adjacent to building material or the ground plane of the lattice (i.e., waspmites cannot "fly"). After moving, each agent is given the opportunity to place a block of building material in one of the six adjacent locations, again excluding any that contain building material. Each block of building

material releases a simulated pheromone specific to its type. All pheromones diffuse through the lattice, but, like agents, they are unable to pass through building material.

8.1.1 Movement

Whereas, previously, wasp agents moved at random through their world, waspmite movement is influenced by locally perceived pheromone gradients. In addition to guiding its construction behaviour, a waspmite control algorithm specifies a weight, w_i, for each of the pheromones that it may encounter. These weights each specify the degree to which the waspmite is attracted towards one of these pheromones.

The probability that a waspmite choses to move to an adjacent empty cell, c, is determined as:

$$p_c \propto \sum_{i}^{T} w_i P_{ic}$$

Where p_c is the probability of choosing to move to location c, w_i is the agent's subjective attraction to the i^{th} pheromone of T distinct pheromone types, and P_{ic} is the intensity of pheromone i at location c.

8.1.2 Building

As before, whether or not an agent decides to place a block of building material, and if so which type to place, is determined by an ordered list of microrules. Each microrule comprises a triggering condition in the form of a particular configuration of block types that, if encountered, will activate

the rule, and a block type that the agent will place if the rule is activated. Additionaly, each microrule may only fire if the local pheromone intensities meet the microrule's conditions. For instance, a microrule might specify that block type A is to be placed adjacent to block type B if and only if there are less than 2 units of pheromone C present, or if the level of pheromone C is lower than that of pheromone D.

8.1.3 Pheromones

Pheromones diffuse and decay at each time step at rates determined by a pair of pheromone-type-specific parameters α_i and β_i, respectively.

When a piece of building material of type i is placed by a waspmite, it contains an initial finite volume of pheromone, I_i. At each time step a constant proportion, α_i, of this pheromone spreads to the surrounding locations through diffusion. Note that pheromone cannot diffuse through building material. Moreover, at each time step, some proportion, β_i, of pheromone is denatured through processes of decay.

The rate of diffusion between two lattice cells that share a common face is therefore proportional to the pheromone gradient between them. If the volume of a specific pheromone at a particular location is x and the volume of the same pheromone at one of its diffusion neighbours is y then the change of pheromone at x may be expressed as $\partial P_x/\partial t = -\alpha(x-y)$, where $0 \leq \alpha \leq \frac{1}{6}$ in order that diffusion remains conservative.

8.2 Waspmite Constructions

The waspmite scheme outlined above subsumes the two prior swarm constructions algorithms that inspired it as special cases. Waspmites are able to implement the paper wasp algorithm [47] by neglecting to employ pheromones. The same waspmites are able to perfectly emulate the artificial termites simulated in [32, 33] by employing only microrules that pay no attention to the local configuration of block types. Consequently the new scheme is capable of generating the union of the classes of structures achievable with each of the original schemes, i.e., any assembly of contiguous blocks, plus a range of conic and spherical solids. How readily the waspmite scheme generates particular examples of these architectures is less easy to determine analytically. Here we explore the performance of the new scheme on a range of simple structures.

[Figure 6 about here.]

Fig 6a depicts a section through one of the simplest waspmite structrures, a hollow hemisphere. Pheromone diffuses at a constant rate from a single seed block, S, placed on the ground-plane of the lattice. Each waspmite moves at random and obeys a single construction rule: place a block of type A in any legal location where the pheromone level lies within a critical range. Since locations exhibiting pheromone intensity within this range are found at a characteristic distance form the seed block, building activity constructs a hollow shell around S. The radius and width of the constructed hemisphere can be altered by changing the diffusion and/or decay parameters of the pheromone, or the lower and/or upper bounds of the rule's

triggering condition. Construction can be accelerated by specifying that blocks of type A release a second type of pheromone that decays relatively rapidly, and encouraging waspmites to climb any locally sensed gradient in this new pheromone. This results in waspmites moving rapidly towards sites of recent building activity. These sites tend to offer more opportunities for further construction, generating more A-pheromone, and thereby recuiting more builders, and so on.

Fig 6b indicates how multiple sources of pheromone can be used to direct waspmites to build more complex conic structures. The intersection between two distinct pheromone distributions can be used to generate a semi-circular wall if waspmites may build only where the intensity of each pheromone is roughly the same, or a semi-circular arch if building is triggered when both pheromones are close to some specific intensity value. This approach is analogous to that used within some developmental biological systems and copied by amorphous computing and spatial computing systems [19].

Fig 6c shows the construction of a simple column. A single seed block, S, releases a pheromone, P, at constrant rate, establishing a hemispherical pheromone distribution. Waspmites follow three rules: 1. place A on top of S; 2. place A on top of A if $P < t$; 3. place B on top of A where $P \geq t$. As with the hemisphere in Fig 6a, the column's height is determined by a combination of the diffusion and decay parameter's of P and the threshold, t.

[Figure 7 about here.]

Fig 7 depicts four stages in the construction of a horizontal square frame constructed to surround a single seed block located on the ground plane. Initially, the seed block establishes a pheromone template by releasing pheromone at a constant rate. One rule establishes A blocks at the centre of the North and South sides of the frame. The rule is triggered when a location is discovered with a particular intensity of pheromone, and roughly equivalent intensities in the two immediately adjacent locations to the East and West. A similar rule establishes blocks of type C at the centre of each of the remaining two sides of the frame (Fig 7a). A second pair of rules establish blocks of type B either side of A-blocks. Analogous rules achieve equivalent placement for blocks of type D and C (Fig 7b). Finally, a pair of rules place further blocks of type B alongside existing B-blocks, while the pheromone intensity remains below some threshold t chosen to correspond to the corner locations. Again, an analogous pair of rules complete the East and West walls (Fig 7c). The type of block placed at each corner is determined circumstantially by whether the triggering condition for a B- or D-block is encountered first at each of the corner locations (Fig 7d). These five pairs of rules were chosen with clarity of exposition in mind. An identical four-sided frame can be achieved with fewer rules if blocks labelled B and A (and C and D) are indistinguishable to waspmites. Again the dimensions of the frame can be scaled by varying the pheromone diffusion and decay parameters.

[Figure 8 about here.]

Fig 8 shows how such a frame can be used as the basis for a hollow cube structure, while Fig 9 demonstrates a combination of the basic column and

arch structures described above. Here the capstone on top of each column releases a pheromone that is used as a template to guide the construction of an arch linking the nearest pair of columns. The way in which different block types interleave at the intersection between each arch is determined circumstantially by which triggering conditions are encountered by the waspmites as they crawl over the partially built construction.

[Figure 9 about here.]

What these results convey is the relatively small number of rules and block types required to construct compound structures, and, by contrast with the original paper wasp scheme, the relative ease with which the size of these architectures can be rescaled. This improved performance resembles that of positional information schemes explored within models of artificial development [19]. We do not claim that the waspmite scheme is able to readily construct all architectures, or even all interesting or useful architectures. Rather, we use this novel scheme as a way of demonstrating that changes to a bio-inspired algorithm can bring about significant changes in performance that are independent of the scheme's competence.

Here we have restricted ourselves to considering hand-designed sets of micro-rules in the spirit of [47]. While algorithms capable of generating the structures presented here were relatively easy to design, a number of challenges were encountered that will require further work. First, it is often easy to see how to achieve a structure in stages, with waspmites first concentrating on the completion of one set of sub-structures before commencing work on the next stage. In [47] these stages are sometimes manually imposed with

a separate set of micro-rules being employed at each stage. Ideally, swarm construction would be able to do without these pre-timed stages. However, general schemes for avoiding the timing issues involved in the construction of complex architectures are not yet developed. Finally, in further work, we intend to explore the automatic generation of waspmite algorithms using a search algorithm. This will require defining (i) a space of waspmite algorithms and (ii) search operators that determine the neighbourhood structure over the space. It remains to be seen whether effective choices for (i) and (ii) can be made.

9 Conclusions

We have argued that bio-inspired algorithms will tend to have limited *performance* even if they have general *competence*, since the biological systems that inspired them were evolved to achieve species-specific objectives. This will be true even where such an algorithm is demonstrably universal, such as the paper wasp collective consturction algorithm explored and extended here.

While the wasp-inspired algorithm might be assumed to have rather wide utility on the basis of its generality, this is in fact undermined by significant limits on its performance. The extended analysis presented here highlights that knowledge of the *performance* of the algorithm is much more important than knowledge of either its ultimate *competence*, or the performance of the biological systems that inspired it. The arguments presented here suggest that this will be true for bio-inspired algorithms in general. Conversely, it

is important to note that an algorithm subject to such strong limitations is still perfectly capable of generating some (nest-like) repeated structures quite robustly, as demonstrated by Théraulaz and Bonabeau [47]. As such, the discovery of even very serious constraints on performance does not necessarily damage the utility of a scheme—in fact, knowledge of such constraints can be leveraged much more readily than any knowledge of the algorithm's generality of competence.

If we limit our ambition to the construction of architectures comprising a single set of contiguous structures, then extending the wasp-inspired construction scheme to incorporate pheromone-mediated behaviour achieved no change in competence. Any contiguous structure buildable by the new waspmite scheme is also buildable under the old scheme. However, we have made a significant difference to the performance of the scheme since waspmites are readily able to generate architectures with long-range structure that would have required a prohibitive number of microrules and block types under the old scheme. Further work will explore the application of evolutionary design techniques to waspmite algorithms in order to automate the discovery of useful architectures.

In one sense, the issue being explored here is not a new one. Initial attempts at *general-purpose* rational AI based on reasoning and logic, e.g., [40], were superseded by expert systems that incorporated and exploited domain-specific knowledge in order to achieve much more impressive performance within a single individual problem domain, e.g., [35]. But interest in these systems itself waned as it became clear how hard it was to discover, represent and exploit such domain-specific knowledge for many kinds of im-

portant problem, e.g., real-time autonomous interaction within an uncertain world such as that described by [10].

In fact, what is desirable in a scheme is neither universality, *per se*, nor evidence of specific performance on a finite set of test problems, since neither may translate into efficient performance across the actual space of current and future problems of interest. Truly useful and impressive schemes will be ones that can convincingly be demonstrated to be easily tailored to solve a *class* of problems that we want solved. In order to demonstrate that this is true for some scheme, one of the two things that we need to understand is *what kind of problems is the scheme suited to* (its domain of performance, or niche). It is clear that neither proofs of general competence nor demonstrations of impressive performance on problem instances constitute this kind of understanding.

However, a second, more fundamental, kind of understanding must also be achieved: we need to be able to characterise the kind of problems that we are interested in solving (their common structural properties). It is precisely when this information is lacking that there is little option but to appeal to a scheme's universality of competence and/or its impressive performance on test problems. It is only with this knowledge of problem structure in hand that we are in a position to discover or design a scheme that is good for a problem domain. There has been some work in this vein at the periphery of AI, often with a focus on natural or cognitive organisation [45, 34, 2], but there has been less interest in this question within the bio-inspired computing community, but see, e.g., [50, 51].

Notice that it may even be the case that a provably non-universal class

of algorithm, such as the perceptron, might be a more desirable scheme than a provably universal one, since it might happen to be suited to a class of problem that we need to solve, despite being provably unable to solve alternative classes of problem.[5] Even where we have no prior expectations of a particular scheme, this issue still bites. For instance, like Beer and Gallagher [4], we may wish to use a type of CTRNN as a control architecture for an artificial agent, without biasing the agent's behaviour in any particular direction. It is still the case that a mere proof of universal competence (such as Funahashi and Nakamura's, 1993, proof that CTRNNs can, in principle, approximate the dynamics of any system to an arbitrary degree of accuracy) is not sufficient to demonstrate that the actual *performance* of the agents will not be biased by the choice of control architecture in some potentially important respect.

In conclusion, niche-centric thinking of the kind argued for in this paper encourages us to *characterise*, rather than merely quantify, an algorithm's performance (as opposed to its its competence or scope), and to compare this characterisation with that of a domain of problems to be solved rather than either problem instances or the space of all problems. This type of thinking implies a reconsideration of the typical working methodology of bio-inspired computing researchers. Only once we accept that, in general, biological devices, processes and organisations are properly viewed as specific to their particular niches, and (in collaboration with biologists) develop

[5]To take this approach is to interpret devices in terms of their *ecological rationality*, that is to interpret them against externalist norms of performance under normal conditions within typical environments, rather than internalist norms of logical consistency, transitivity, completeness, monotonicity, etc. [13].

theoretical accounts of what it is that individual biological devices, processes or organisations are good at—what it is that they have been "designed" to achieve—will we be in a position to exploit idealisations of them efficiently.

Acknowledgements

Thanks to Jason Noble, Richard Watson and the Biosystems and SENSe research groups at Leeds and Southampton, respectively, for discussion of these ideas. Seth Bullock acknowledges the support of EPSRC (UK) grant no EP/C51632X/1, and also the World-wide Universities Network and the generosity of the University of California San Diego during a research visit to its Science Studies Program. We also acknowledge the support of two Nuffield Undergraduate Research Bursaries.

References

[1] J. Beal and J. Bachrach. Infrastructure for engineered emergence on sensor/actuator networks. *IEEE Intelligent Systems*, 21(2):10–19, 2006.

[2] W. Bechtel and R. C. Richardson. *Discovering Complexity: Decomposition and Localization as Strategies in Scientific Rresearch*. Princeton University Press, Princeton, 1993.

[3] R. D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3:469–509, 1995.

[4] R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1:91–122, 1992.

[5] E. W. Bonabeau, M. Dorigo, and G. Théraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Oxford, NY, USA, 1999.

[6] E. W. Bonabeau, M. Dorigo, and G. Théraulaz. Inspiration for optimization from social insect behaviour. *Nature*, 406:39–42, 2000.

[7] E. W. Bonabeau, S. Guérin, D. Snyers, P. Kuntz, and G. Théraulaz. Three-dimensional architectures grown by simple 'stigmergic' agents. *BioSystems*, 56:13–32, 2000.

[8] E. W. Bonabeau, G. Théraulaz, J.-L. Deneubourg, N. R. Franks, O. Rafelsberger, J.-L. Joly, and S. Blanco. A model for the emergence of pillars, walls and royal chambers in termite nests. *Philosophical Transactions of the Royal Society of London, Series B*, 353:1561–1576, 1997.

[9] J. Bowlby. *Attachment*. Hogarth, London, 1969.

[10] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[11] O. H. Bruinsma. *An analysis of building behaviour of the termite Macrotermes subhyalinas (Rambur)*. PhD thesis, Landbouwhogeschool, Wageningen, Netherlands, 1979.

[12] C. L. Buckley. *A systemic analysis of the ideas imminent in neuromodulation*. PhD thesis, School of Electronics and Computer Science, University of Southampton, UK, 2007.

[13] S. Bullock and P. M. Todd. Made to measure: Ecological rationality in structured environments. *Minds and Machines*, 9(4):497–541, 1999.

[14] J.C. Chaplin, N.A. Russell, and N. Krasnogor. Implementing conventional logic unconventionally: Photochromic molecular populations as registers and logic gates. *Biosystems*, (in press), 2012.

[15] N. Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA, 1965.

[16] D. Cliff and S. Bullock. Adding 'foveal vision' to Wilson's animat. *Adaptive Behaviour*, 2(1):49–72, 1993.

[17] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.

[18] R. Dawkins. *The Extended Phenotype: The Long Reach of the Gene*. Oxford University Press, 1982.

[19] R. Doursat. Programmable architectures that are complex and self-organized: From morphogenesis to engineering. In S. Bullock, J. Noble, R. A. Watson, and M. A. Bedau, editors, *Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems*, pages 181–188. MIT Press, Cambridge, MA, 2008.

[20] H. A. Downing and R. L. Jeanne. Nest construction by paper wasps, polistes: a test of stigmergy theory. *Animal Behaviour*, 36:1729–1739, 1988.

[21] P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life*, pages 205–213. MIT Press, Cambridge, MA, 1997.

[22] R. A. Foley. The adaptive legacy of human evolution: A search for the environment of evolutionary adaptedness. *Evolutionary Anthropology*, 4:194–203, 1997.

[23] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.

[24] P. Funes and J. Pollack. Computer evolution of buildable objects. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 358–367. MIT press, Cambridge, MA, 1997.

[25] J.-L. Giavitto and A. Spicher. Topological rewriting and the geometrization of programming. *Physica D*, 237(9):1302–1314, 2008.

[26] G. Hornby and J. Pollack. The advantages of generative grammatical encodings for physical design. In *Congress on Evolutionary Computation*, pages 600–607. IEEE Press, Piscataway, NJ, 2001.

[27] T. G. Howsman, D. O'Neil, and M. A. Craft. A stigmergic cooperative multi-robot control architecture. In J. Pollock, M. Bedau, P. Husbands, T. Ikegami, and R. A. Watson, editors, *Ninth International Conference on Artificial Life*, pages 88–93. MIT Press, Cambridge, MA, 2004.

[28] T. Huntsberger, G. Rodriguez, and P. Schenker. Robotic challeges for robotic and human mars exploration. In W. C. Stone, editor, *Fourth International Conference on Robotics for Challenging Situations and Environments*, pages 84–90. American Society of Civil Engineers, 2000.

[29] R. L. Jeanne. The adaptiveness of social wasp nest architectures. *The Quarterly Review of Biology*, 50:267–287, 1975.

[30] D. Kahnemann, P. Slovic, and A. Tversky. *Judgement Under Uncertainty: Heuristics and Biases*. Cambridge University Press, Cambridge, 1982.

[31] S. Kumar and P. J. Bentley. Biologically plausible evolutionary development. In A. Tyrrell, P. Haddow, and J. Torresen, editors, *Proceedings of the Fifth International Conference on Evolvable Systems: From Biology to Hardware*, pages 57–68. Springer, 2003.

[32] D. Ladley and S. Bullock. Logistic constraints on 3D termite construction. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 178–189. Springer-Verlag, Berlin, 2004.

[33] D. Ladley and S. Bullock. The role of logistic constraints in termite construction of chambers and tunnels. *Journal of Theoretical Biology*, 234:551–564, 2005.

[34] R. Levins. Complex systems. In C. H. Waddington, editor, *Towards*

a Theoretical Biology, pages 73–88. Edinburgh University Press, Edinburgh, 1970.

[35] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg. *Applications of Artificial Intelligence for Organic Chemistry: The* DENDRAL *Project*. McGraw-Hill Book Company, 1980.

[36] R. G. Millikan. *Language, Thought and Other Biological Categories*. MIT Press, Cambridge, MA, 1984.

[37] R. G. Millikan. *White Queen Psychology and Other Essays for Alice*. MIT Press, Cambridge, MA, 1993.

[38] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[39] R. Nagpal. Programmable self-assembly using biologically-inspired multi-agent control. In *First International Conference on Autonomous Agents*. Bologna, July 15-19, 2002.

[40] A. Newell and H. Simon. GPS: A program that simulates human thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 279–296. McGraw-Hill, New York, NY, 1963.

[41] P. Prusinkiewicz and A. Lindenmeyer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[42] J. Reinitz, E. Mjolsness, and D. H. Sharp. Model for cooperative control of positional information in drosophila by bicoid and maternal hunchback. *Journal of Experiemental Zoology*, 271:47–56, 1992.

[43] M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, Cambridge, MA, 1994.

[44] I. Salazar-Ciudad, J. Garcia-Fernández, and R. Solé. Gene networks capable of pattern formation. *Journal of Theoretical Biology*, 205:587–603, 2000.

[45] H. A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.

[46] A. O. Tarakanov, V. A. Skormin, and S. P. Sokolova. *Immunocomputing: Principles and Applications*. Springer-Verlag, Heidelberg, 2003.

[47] G. Théraulaz and E. W. Bonabeau. Modelling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology*, 177:381–400, 1995.

[48] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[49] J. S. Turner. Architecture and morphogenesis in the mound of *Macrotermes michaelseni* (sjöstedt) (isoptera: Termitidae, macrotermitinae) in northern namibia. *Cimbebasi*, 16:143–175, 2000.

[50] R. A. Watson. *Compositional evolution: Interdisciplinary investigations in evolvability, modularity, and symbiosis*. PhD thesis, Brandeis University, MA. USA, 2002.

[51] R. A. Watson and J. B. Pollack. Modular interdependency in complex dynamical systems. *Artificial Life*, 11(4):445–457, 2005.

[52] J. Wawerla, G. S. Sukhatme, and M. J. Matarić. Collective construction with multiple robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2696–2701, 2002.

[53] J. Werfel and R. Nagpal. Extended stigmergy in collective construction. *IEEE Intelligent Systems*, 21(2):20–28, 2006.

[54] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Sante Fe, NM, USA, 1995.

[55] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.

[56] L. Wolpert. Positional information and the spatial pattern of cellular differentiation development. *Journal of Theoretical Biology*, 25:1–47, 1969.

[57] L. Wolpert. Positional information revisited. *Development*, 107:3–12, 1989.

List of Figures

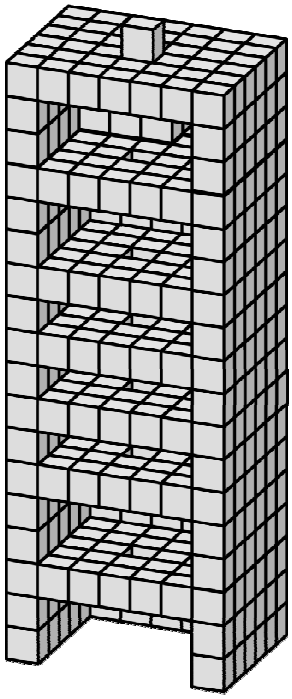

Figure 1: A layered structure resembling an idealised wasp nest generated using the rule-set detailed by Théraulaz and Bonabeau [47].

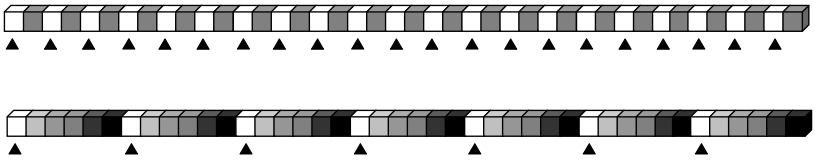

Figure 2: Perfect solutions to the problem posed by fitness function F_1 with (top) $n = 1$, $T = 2$ and (bottom) $n = 5$, $T = 6$, achieved after 500 generations of evolutionary search. Arrowheads indicate blocks that contribute positively to fitness. Structures are built from left to right, but measured from right to left for the purposes of assigning fitness.

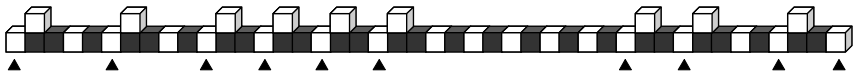

Figure 3: The best structure discovered after 500 generations of evolutionary search for the problem posed by fitness F_1 with $n = 2$, $T = 2$. Arrowheads indicate blocks that contribute positively to fitness.

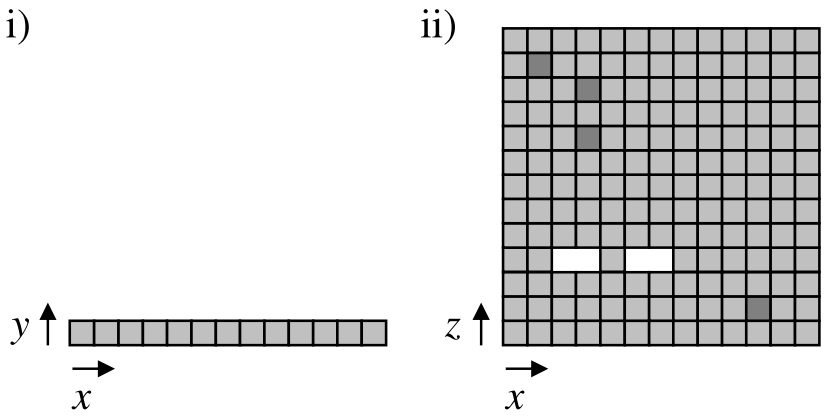

Figure 4: (i) An elevation and (ii) a plan of the best structure discovered after 500 generations of evolutionary search for the problem posed by fitness function F_2 with $T = 2$.

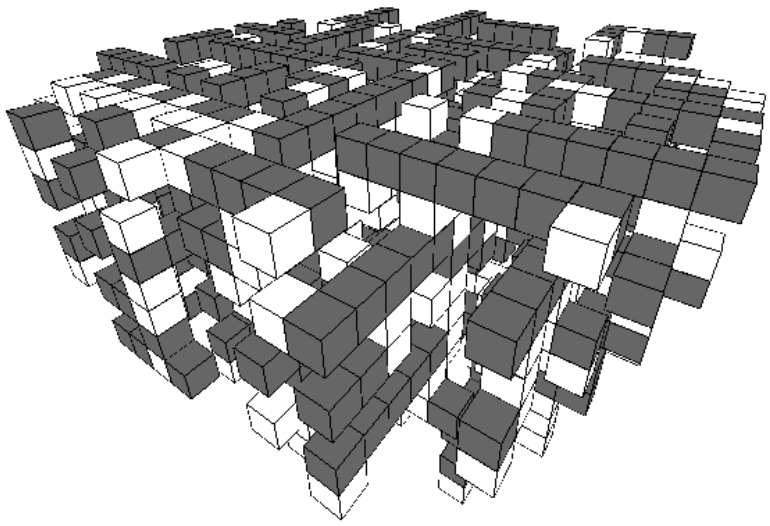

Figure 5: The best structure discovered after 500 generations of evolutionary search for the problem posed by fitness function F_3 with $T = 2$.

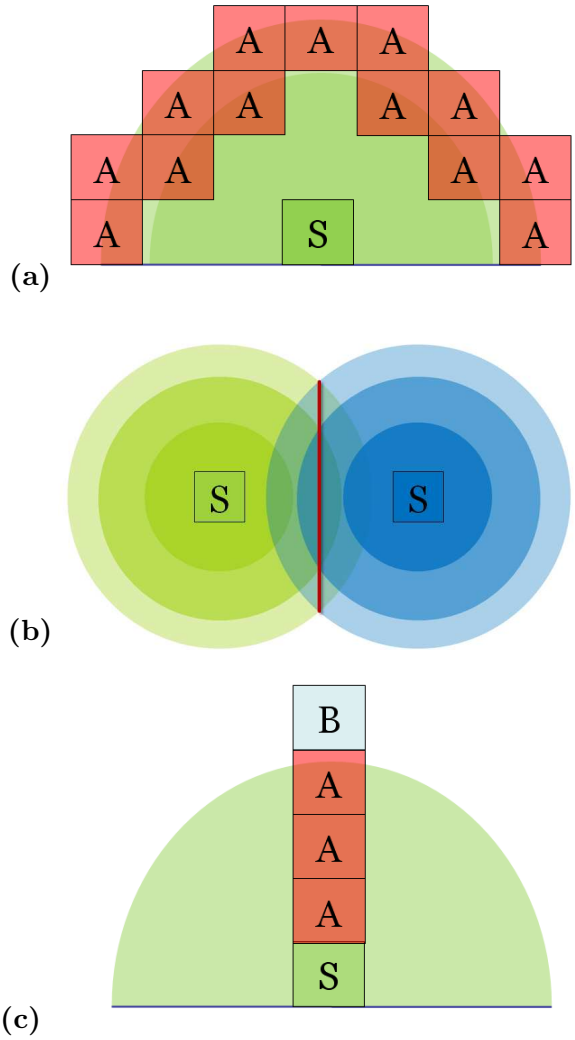

Figure 6: Schematics depicting three simple waspmite structures: (a) a section through a hemispherical structure built around a single seed block, S, placed on the ground-plane of the lattice, (b) an arch or semi-circular wall built mid-way between two seed blocks releasing distinct pheromones at a constant rate, and (c) a column of A-blocks erected on top of a single seed block, S, and capped with a final block of type B. See text for details.

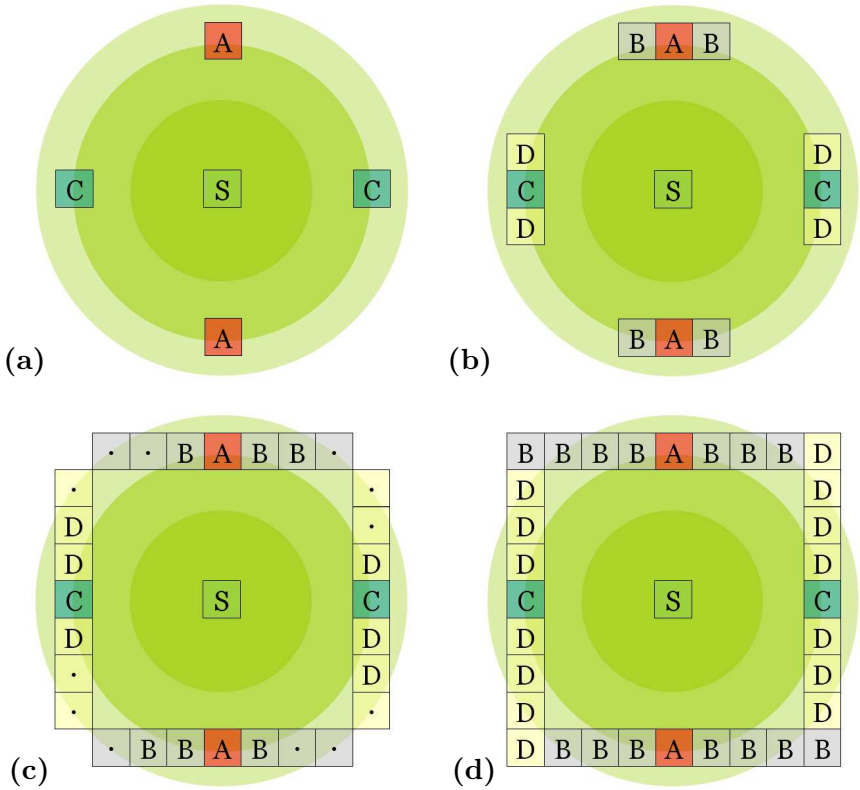

Figure 7: Schematics depicting four stages in the construction of a horizontal square frame surrounding a single seed block, S, located on the ground plane. See text for details.

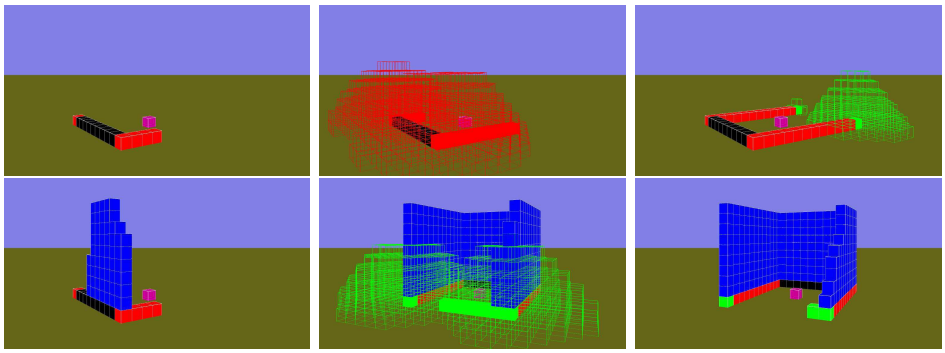

Figure 8: Stages in the formation of a square frame (top row), and a hollow cube (bottom row). In both cases building is initiated by the placement of a single block (depicted in magenta) in the centre of the ground plane. Distinct types of building material are represented by solid cubes of different colours. Distributions of distinct types of pheromone are indicated by wire-frame cubes of different colours. Waspmite agents are not depicted.

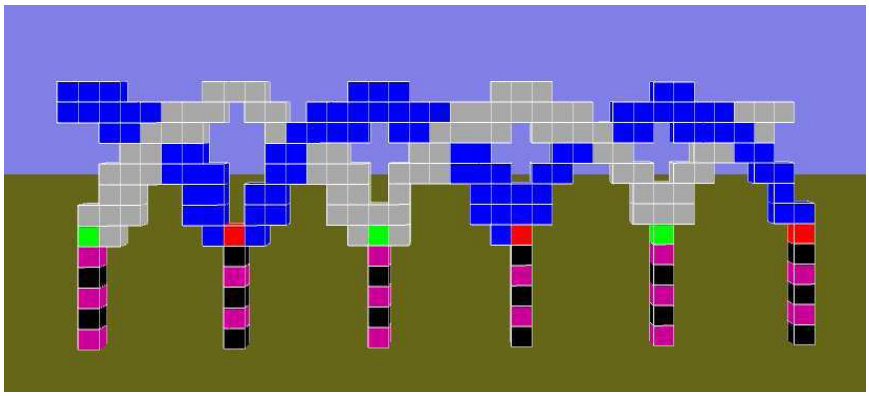

Figure 9: A series of interleaving arches mounted on a row of columns.