Cooperation Strategies for Swarms of Collaborating Robots: Analysis of Time-Stepped and Multi-Threaded Simulations

Liam McGuigan, Catherine Saunders, Roy Sterritt, George Wilkie
School of Computing, Faculty of Computing, Engineering and the Built Environment
Ulster University
Jordanstown, N. Ireland
email: mcguigan-l8@ulster.ac.uk, c.saunders@ulster.ac.uk, r.sterritt@ulster.ac.uk, fg.wilkie@ulster.ac.uk

***Abstract*— Swarms of robots have been proposed for use in many tasks, such as space exploration, search & rescue operations, and mine clearance. For a robot swarm to be successful, it needs to be self-adaptive, making its own decisions and adjusting its behaviour without relying on human intervention. This paper investigates the potential for using an autonomic system for a robot swarm engaged in a foraging task, capable of adjusting its cooperation strategy based on the ongoing performance in the task, rather than sticking with an initial strategy. The results show that while support for changing the strategy completely is limited, there remains the potential for adjusting the parameters of the given strategy to suit the ongoing situation. In addition, a comparison of two approaches to the implementation of a simulation is also presented. A time-stepped approach is compared with the multi-threaded approach used in previous work, with a view to embedding simulation within the swarm as a means of aiding the autonomic decision-making process through simulation of potential options. It is found that even when the underlying robot behaviour is identical, the time-stepped simulation is faster and more flexible, and is therefore more suitable for embedding.**

Keywords- Swarm robotics; Self-adaptation; Autonomic Computing; Simulation.

I. INTRODUCTION

This paper is an extended version of the work published in [1]. It extends those results with new findings from further research.

The use of robotic swarms consisting of a large number of robots operating in concert can benefit applications, such as space exploration [2][3], search & rescue [4] and mine clearance [4][5] among others, taking advantage of a robot's ability to operate in conditions where human involvement is too dangerous or difficult.

The individual craft in a robotic swarm will need to be capable of managing themselves without requiring constant supervision. They may be required to make quick decisions to protect themselves or to act on opportunities, and will need to adapt to best suit the conditions of the task being carried out [6]. This can be achieved by making the swarms autonomic.

Autonomic computing concepts will embody the swarm with the properties of self-configuration, self-healing, self-optimization and self-protection, ensuring that the swarm [7] is implemented by including an autonomic component running a Monitor, Analyse, Plan, Execute, with a shared Knowledge (MAPE-K) control loop to monitor and analyse the situation, and plan and execute any changes to behaviour aided by a knowledge base of pre-set or previously acquired information [8], as seen in Figure 1. Autonomic robotics combines the concepts of MAPE-K from autonomic computing, with Intelligent Machine Design (IMD) from robotics [9][10].

Due to the cost and impracticality of using real hardware during the development of large-scale swarm behaviour, simulators are often used in the process [11], able to create artificial swarms of hundreds or even thousands of robots engaged in tasks, such as foraging, surveillance and exploration of unknown environments.

The research presented in this paper has two objectives. The first is to investigate the potential for self-adaptation through selection of a cooperation strategy during a foraging task, through analysis of the performance of different strategies over the course of the task. The second objective is to identify which of two simulation implementation approaches used would be most suitable for deploying on an individual agent within the swarm as a means of using simulation-in-the-loop to help with the in-task strategy switching decision.

The rest of this paper is organised as follows. Section II discusses related work on self-adaptation in swarm robotics, and the varying use of simulations in development. Section III describes the foraging task used in the simulation, and the

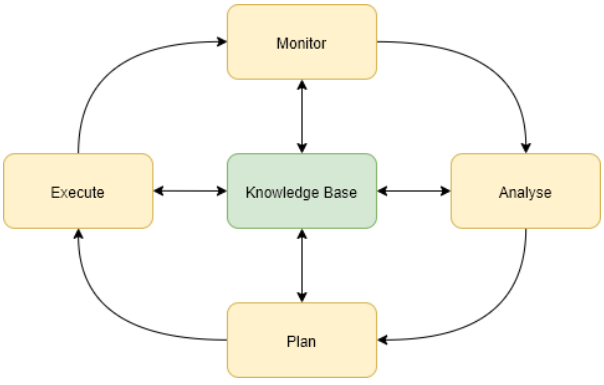


Figure 1. MAPE-K loop, as used by an autonomic component. The loop proceeds through each of the four stages in turn.

general robot behaviour. Section IV describes the cooperation strategies used in the research, while Section V describes the simulators used for comparison. Section VI describes the test scenarios that were run, Section VII presents the results of the cooperation strategy comparison, and Section VIII presents the results of the simulator comparison. Section IX concludes the paper with a summary and presents the future research directions.

II. Related Work

The following subsections discuss current research in swarm self-adaptation, and the use of simulations within swarm development.

A. Swarm Self-Adaptation

Self-adaptation of a robotic swarm concerns the ability of the swarm to adjust its behaviour in response to external or internal conditions, such as a foraging swarm choosing to abandon a depleted deposit in order to find newer deposits, or a surveillance swarm organizing itself so as to provide maximum coverage of the target area.

Swarm self-adaptation can be considered based on two factors – the approach to adaptation, and the location within the swarm where this is applied. Approaches to swarm adaptation include engineering emergent processes where adaptation arises naturally out of the agent behaviour [12], reasoning and learning approaches where the swarm explicitly reasons about the decision being made [13] and may learn from experience [14], and evolutionary approaches which explore alternatives through genetic algorithms [15].

Regarding location, a lot of the research focuses on applying adaptation to individual agent behaviour [16]–[18]. This low-level adaptation results in a bottom-up approach to swarm behaviour, with the resulting performance of the swarm arising from the aggregate performance of the individual agents. This can allow for more specific adaptation, such as balancing an individual's conflicting objectives [19], which may be difficult to apply at the swarm level. Agent behaviour adaptation can have the most direct impact on the swarm's performance, but it is difficult for an agent to make an individual decision on aspects of collaboration or coordination between multiple agents.

Adaptation through the selection of swarm-level cooperation strategies can be used to address the problem of collaboration. In this approach, agents within the swarm can collectively determine an alternative approach which is swapped with the existing agent behaviour either in part or in whole. This selection may be driven by an autonomic component that assesses the suitability of alternative strategies [14][20], and may be employed with in a subset of the entire swarm [21].

Where research has traditionally considered homogeneous swarms with simpler behaviour, a swarm of heterogeneous robots can be more flexible and capable of handling a wider variety of situations [22].

This research is focused on identifying the potential for swarm-level adaptational changes by assessing the performance of a selection of candidate strategies in a set of scenarios. Through noting any effect the scenario has on the performance of a particular strategy, the benefits of the ability to select an alternative strategy will become apparent.

B. Use of Simulations in Swarm Development

Simulation has long been employed as a tool for the development of robotic and swarm simulations, providing the means to test and analyse systems in an artificial environment. Simulators range in complexity, from detailed physical simulations of actual robots [11][23], to abstract approaches where robots move within a grid-based environment. The difficulty of producing an accurate simulation of the real world can manifest as the "reality gap" [24], where results obtained in simulation are not replicated in reality. Nonetheless, it is not necessary for the results of a simulation to be precisely reproducible in the real world for the simulation itself to prove useful.

Simulation need not be restricted to the offline development phase. It may be used to assist the decision making process [25][26], trying out "what-if" scenarios in order to assess the effects of potential actions or strategies ahead of time. For this to be effective, a simulation must be detailed enough to provide useful information, while remaining lightweight enough to be able to run on an individual agent within the swarm.

When choosing or designing a simulator for researching robotic swarms, the accuracy of the physical simulation required will depend on the impact specific hardware has on the research being conducted. Developing a robotic controller without a suitably accurate physical simulation can lead to the robots in the simulation carrying out behaviour that is impossible with the actual robots [27], but when researching purely software based systems, abstractions can be used to trade accuracy for a faster execution time [23].

Further gains in execution time may be made by simplifying the world representation. Grid based approaches need not produce markedly different results to continuous space [28], and can be used in cases where the specific motion of agents can be abstracted.

The majority of multi-robot simulators available make use of discrete time when updating their simulations, in which all agents and physical reactions are updated in sequence with a small time step, rather than independent execution in real time, such as assigning a robot its own execution thread. This ensures synchronous execution of robots [23] and simplifies physical interactions.

The research conducted in this paper abstracts physical movement using cell-based movement within a grid, and the performance is contrasted with a real-time, multi-threaded approach used in previous research [20].

III. FORAGING TASK DESCRIPTION

This research focuses on simulations of a swarm of heterogeneous robots engaged in a foraging task. Each run begins with a world constructed as a rectangular grid of cells, in which a number of target items and robots are initially placed, as shown in Figure 2. Each cell may contain only one item, but any number of robots. It can be assumed that each cell is represents a larger area than the footprint of a single robot, and the simulation may therefore ignore potential collisions.

The robots are tasked with foraging for all the items in the area. Each item is associated with a type, and may only be foraged by a robot with the corresponding type. The simulation proceeds until all the items have been foraged.

Foraging of an item happens at the place it is found, rather than returning to a home base – the simulation can therefore be considered to logically represent applications, such as mine deactivation, analysis of mineral deposits, or environmental cleanup.

Each robot begins in an exploration state, during which they move towards a random location in the world. Each step, if an item is found, the robot will forage if it is able to, removing the item from the world, before resuming exploration.

In the extended research, the behaviour in the exploration state has been modified so that instead of selecting a random location on the map to move towards over time, a robot engages in a random wander, selecting an available direction to move each update.

IV. COOPERATION STRATEGIES

For the initial research, cooperation during the task is determined using one of three strategies, as developed in [20]. The extended research updated this with a set of three configurable strategies, two of which replicate the initial approaches. The following subsections describe each set of strategies.

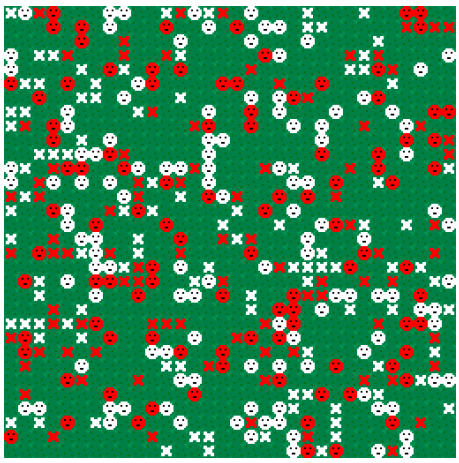


Figure 2. A view of the simulation at the start of a task. The colour of a robot (face) or item (cross) indicates its type.

A. Initial Strategies

When a robot encounters an item that it cannot forage, it broadcasts a help request with a range of 5 cells. The behaviour of the robots is then determined by the strategy:

1) Multiple Responders: If not already engaged in foraging or responding to a previous request, a receiving robot of the correct type will respond to the request by transitioning to a Respond state, in which it moves towards the item. All receivers, whether they can forage the item or not, will rebroadcast the message. In this way, the message will filter throughout the swarm. The robot initiating the help request plays no further part in the cooperation and returns to exploration.

2) Selective Responders: The behaviour here is similar to the Multiple Responders approach, but the message is only rebroadcast if the receiving robot cannot help. This works to reduce the number of robots responding to the request.

3) One Responder: The robot initiating the request transitions to a WaitForOffers state. Offers to help are sent by receiving robots that meet the criteria, who themselves transition to a WaitForAssignment state. No rebroadcasting of the message occurs. If no offers are received after a short delay, the requesting robot returns to its previous behaviour, otherwise it assigns the task to the nearest responding robot and resumes exploration. Robots that do not receive assignment after a period of time return to exploration, while the assigned robot transitions to the Respond state.

Both Multiple and Selective Responder strategies are likely to result in multiple robots moving towards the item. This would provide contingency in the event of robot failure before reaching the target item. Robot failure is not implemented in the current simulation, but will be in a future study.

B. Extended Configurable Strategies

These strategies were used in the extended research. Each strategy offers one or more parameters to fine tune the behaviour.

1) Simple Broadcast: This strategy replicates the behaviour of the Multiple and Selective Responders strategies described above, but parameters allow for the modification of the broadcast range, the number of rebroadcasts, and the likelihood of a robot responding to the request. The parameters are listed in Table I.

2) Help Recruitment: This strategy replicates the One Responder strategy, with parameters to allow adjustment of the broadcast range and maximum number of recruits, as listed in Table II.

3) Blackboard: In this new strategy, each robot maintains a record of items found and items foraged, and periodically synchronises this information with neighbouring robots via broadcasts. During the update loop, a robot will move to the nearest unforaged item of a

matching type that it is aware of, provided it is within the configured range. If a robot subsequently receives information to say that an item has been foraged, it will abandon its efforts to retrieve the item, and resume exploration. The configuration parameters are described in Table III.

This is a fully decentralised implementation in which robots gather and share information. When a robot finds an item, it records this in its memory as unforaged. If it subsequently forages the item, it updates the record. During synchronisation, a robot updates its current memory, adding new items it was not aware of, or updating the forage state of items it previously believed to be unforaged.

Due to the decentralised nature of the blackboard implementation, each robot will have a different interpretation of the current state of the field. However, as the synchronisation process will only move an item from unknown, to found, and then to foraged, it is not possible for the swarm to "forget" an item's state, only to be unaware of any change. As the simulation proceeds, the data will filter throughout the swarm as robots rebroadcast the new details.

TABLE I. SIMPLE BROADCAST PARAMETERS

Parameter	Description	Values
Broadcast Range	Maximum cell distance each broadcast for help can reach	[1 .. ∞]
Max Chain	Number of rebroadcasts for any given message	[0 .. 4]
Rebroadcast Mode	Determines whether or not a robot rebroadcasts	Never Type Mismatch Always
Response Type	How a robot responds to a request	Always Response Curve
Response Distance	The maximum distance at which a robot may respond (Response Curve only)	[0 .. ∞]

TABLE II. HELP RECRUITMENT PARAMETERS

Parameter	Description	Values
Broadcast Range	Maximum cell distance each broadcast for help can reach	[1 .. ∞]
Max Recruits	Maximum number of robots to assign an item to	[1 .. ∞]

TABLE III. BLACKBOARD PARAMETERS

Parameter	Description	Values
Broadcast Range	Maximum cell distance each synchronisation broadcast can reach	[1 .. ∞]
Response Range	Maximum distance of an unforaged item that prompts a response	[1 .. ∞]
Sync Period	Simulation ticks between synchronisations	[1 .. ∞]

V. SIMULATORS

The following sections describe the simulators that were used in this research.

A. Time-Stepped Simulator

In the time-stepped simulator, each robot in the simulation is updated in sequence, with a single tick of the simulation ending once all robots have been updated. Each tick of the simulation can therefore represent a discrete period of time, and the performance of the swarm at completing the task can be measured by the simulation ticks taken to complete the task. Figure 3 shows an example of how the time-stepped simulator updates.

Robots are implemented as a finite state machine (FSM). In each tick of the simulation, the robot will update its current state, and check for state transition conditions. Thus, in a single tick a robot may choose to move one cell, to forage an item underfoot, or to participate in the cooperation strategies described in Section IV.

In this simulation, broadcast messages are first buffered, and only processed after all robots have been updated, in order to maintain synchronisation between independent robot movements, and avoid the possibility of a later-updated robot being able to process a message that was sent in the same tick by an earlier-updated robot. Once the messages are processed and received by reach robot within range, they are further shuffled on the receiving robot in order to remove the effects of update order. If Robot A and Robot B both send a message to Robot C in the same tick, Robot C would always respond to Robot A first. Shuffling the message list before processing removes this problem.

B. Multi-Threaded Simulator

The multi-threaded simulator implementation used here is the same as that presented in [20]. In this implementation, each robot is assigned its own independent CPU thread, complete with pauses to allow for real-time viewing of the simulation.

Each robot executes an infinite loop and carries out a set of steps in sequence: process messages, move one cell towards a target destination, look for an item in the current cell, and forage/request help as required. Figure 4 shows an example of how the multi-threaded simulator operates.

Broadcast messages are sent and buffered by each robot, and processed at the start of their loop. If the robot is using the One Responder strategy, after a help request is sent the robot pauses to wait for an offer, and similarly after an offer is sent, a robot pauses to wait for assignment.

Unlike the time-stepped simulator, this simulator is restricted to a 30x30 map, so all comparisons between the two must be made on that map size.

C. Redesigned Multi-Threaded Simulator

In order to address the fixed map size and differences in robot implementation in the previous multi-threaded simulator, this was rewritten from the ground up in order to

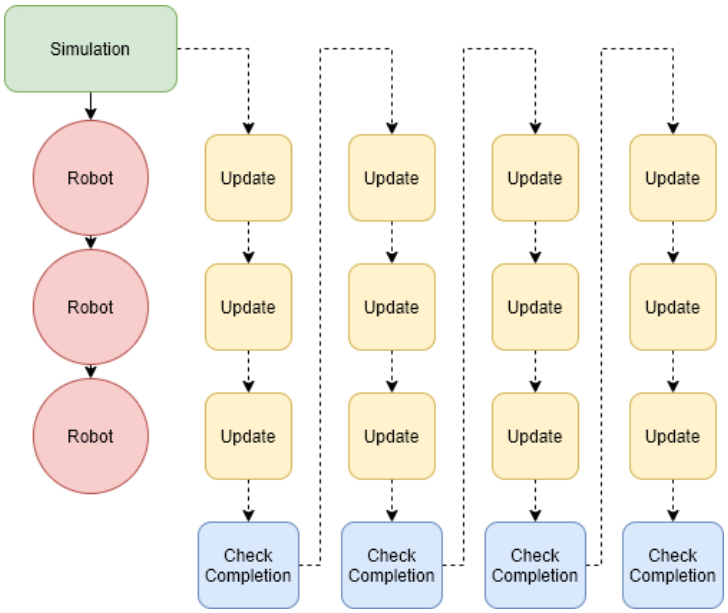


Figure 3. An example of how the time-stepped simulator updates the robots. All robots are updated in sequence on the same thread, represented by the dotted line, before the simulator checks for task completion. If the task is still active, the process repeats.

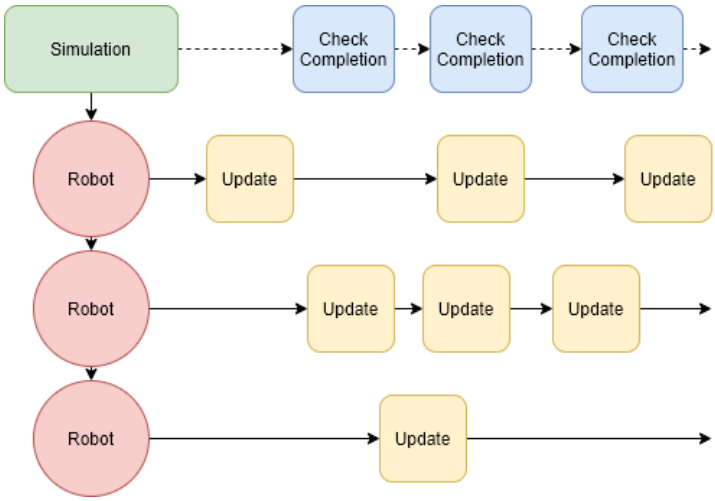


Figure 4. An example of how the multi-threaded simulator updates the robots. Once created, each robot updates on a separate thread to the main simulator and to each other. The frequency and order in which the robots update cannot be controlled. The execution of an update may also be halted at any time – this has not been shown for clarity.

allow a greater variety of tests, and to ensure robot behaviour is identical.

Again, each robot is assigned its own CPU thread and started in a random order, however thread execution is no longer tied to display considerations, and sleeps are limited to 1 millisecond per loop in order to prevent the operating system from locking up during execution. Robots implement the same state machine as in the time-stepped simulator, and messages are also handled in the same way, however no shuffling occurs as the order robots are updated is no longer fixed. If a robot must wait in a particular state, it does so over a number of updates rather than halting execution.

In this implementation, only the Help Recruitment strategy has been implemented, as the intention is to compare performance of the simulators rather than evaluate strategies.

VI. Test Scenarios

The following subsections describe the test scenarios used to evaluate the strategies and simulator approaches.

A. Initial Research

For the initial research, three scenarios were used as described in Table IV.

The Robot Type Imbalance scenario represents a scenario where the swarm configuration deployed is not best suited to the task, and must adapt. The Item Type Imbalance scenario represents one where the reality of the mission differs from that expected, and again the swarm must adapt.

To compare the performance of the cooperation strategies, each scenario is tested on both 30x30 and 90x90 maps. Each simulation is run 30 times, with the initial placement of items and robots randomised at the start of each run.

For simulator comparison, due to limitations of the multi-threaded simulator used, only the Equal Split and Robot Type Imbalance scenarios were able to be run on a 30x30 map. Each scenario was run 30 times on the time-stepped simulator, and 10 times on the multi-threaded simulator.

In assessing the performance of each strategy, the number of simulation ticks until completion of the task is the main metric, as it is a measure of the time taken to forage all items. If the energy cost of actions taken by robots is of interest, then the total number of steps and the number of messages broadcast will also become factors. The simulation does not currently assign an energy cost to individual actions, but the counts may be used as a guide, and for each metric a lower value is considered more efficient.

B. Extended Research

Following on from the above work, two further sets of tests were conducted to compare strategies and simulator approaches. To confirm that no dominant strategy exists and thus support the use of an autonomic system for strategy selection, variants of each of the new set of strategies were created, as described in Table V.

These strategies were each tested in a variety of scenarios created by adjusting the map size, robot counts, item counts, and the ratios of each type, as described in Table VI. A total of 600 scenarios were used to compare the strategies.

To establish any differences in performance between the two simulation implementations, tests were run on four map sizes, each featuring 256 robots and items evenly distributed between the two types, in order to examine the performance of the two simulations in larger environments.

Each scenario was run using the Help Recruitment strategy, and also with no cooperation strategy enabled. To account for the increased distance between robots that would arise in a larger map, the broadcast range for the Help Recruitment strategy was increased in as the map size increased, with 32x32, 64x64, 96x96 and 128x128 maps having ranges of 4, 8, 16 and 24 respectively. Each scenario was run 100 times on each simulator.

For each set of simulations, the average real time to complete the task was recorded, along with the simulation ticks (for the time-stepped simulation) or mean number of robot updates (for the multi-threaded simulation), the total steps taken by all robots in the task, and the number of messages sent, also broken down by message type.

TABLE IV. INITIAL TEST SCENARIOS

Scenario	Items		Robots	
	White	***Red***	***White***	***Red***
Equal Split	100	100	100	100
Robot Type Imbalance	100	100	180	20
Item Type Imbalance	180	20	100	100

TABLE V. STRATEGY VARIANT PARAMETERS

Strategy	Parameter	Values	Variants
Simple Broadcast	Broadcast Range	8, 16	40
	Max Chain	0, 1, 2	
	Rebroadcast Mode	Never, Type Mismatch, Always	
	Response Type	Always, Response Curve	
	Response Distance	32, 64, 128	
Help Recruitment	Broadcast Range	8, 16, 24, 32	12
	Max Recruits	1, 2, 3	
Blackboard	Broadcast Range	8, 16	32
	Response Range	8, 16, 32, 64	
	Sync Period	8, 16, 32, 64	
No Cooperation	N/A		1
Total			**85**

VII. RESULTS – COOPERATION STRATEGY COMPARISONS

The following subsections present the results of the initial research, followed by the extended work which followed.

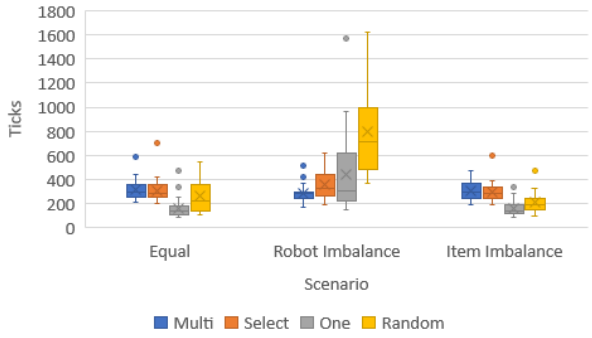
A. Initial Research

Figure 5 shows the ticks to completion, steps taken, and messages sent for each of the test scenarios in a 30x30 grid.

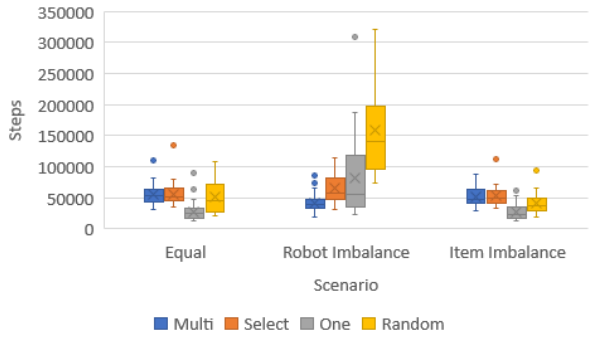
Comparing the results in both the Equal and Item Imbalance strategies, the One Responder strategy is the best performing approach, having the lowest count in each metric. Multiple and Selective Responder strategies can actually perform worse than no cooperation strategy at all, which can be explained by robots that respond to messages halting any exploration while they respond.

TABLE VI. SCENARIO GENERATION PARAMETERS

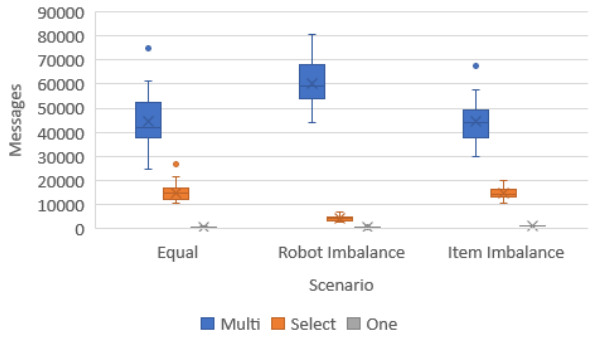
Parameter	Values
Map Size	32x32, 64x64, 96x96, 128x128
Items	64, 128, 256, 384, 512
Item Ratio (White:Red)	1:1, 7:1
Robots	64, 128, 256, 384, 512
Robot Ratio (White:Red)	1:1, 7:1, 1:7



(a) Simulation ticks



(b) Steps taken



(c) Messages sent

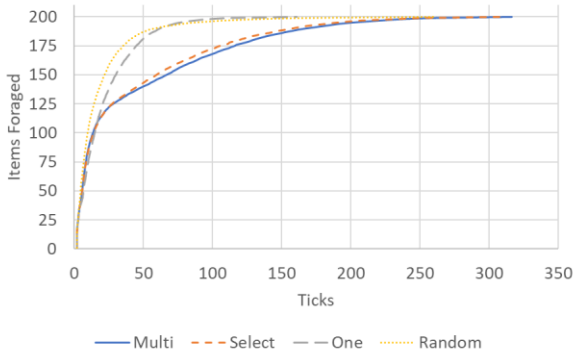
Figure 5. Metrics for each cooperation strategy in a 30x30 map. Circles represent data outliers: (a) simulation ticks, (b) total steps taken, (c) messages sent

In the Robot Imbalance scenario (Figure 5 (b)), however, One Responder does not reliably perform, and is subject to a great deal of variance caused by the initial placement of items and robots, and the subsequent movement of robots within the arena.

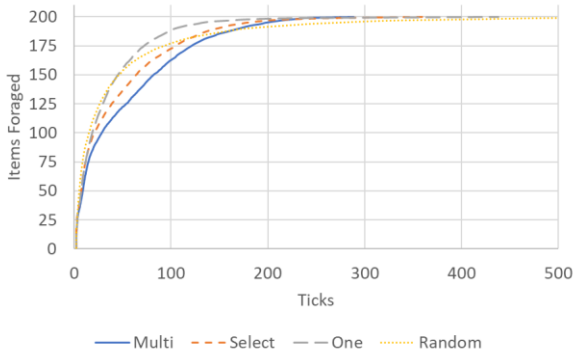
When considering energy costs, Multiple Responders has an extremely high message count setting it apart from Selective Responders, which it otherwise performs very similarly to. A full assessment of the respective efficiency of each would require an assignment of cost to each of the metrics, with the total cost calculated accordingly.

Figure 6 shows the progress of each strategy over time for the three scenarios. In Equal (a) and Item Imbalance (c) scenarios, performance is again similar, however it is notable that using no cooperation strategy at all is the quickest approach until the item count decreases substantially, after which One Responder performs best. This would suggest some system of changing the cooperation strategy used during the test based on the changing situation could lead to stronger overall performance, at least in terms of time taken.

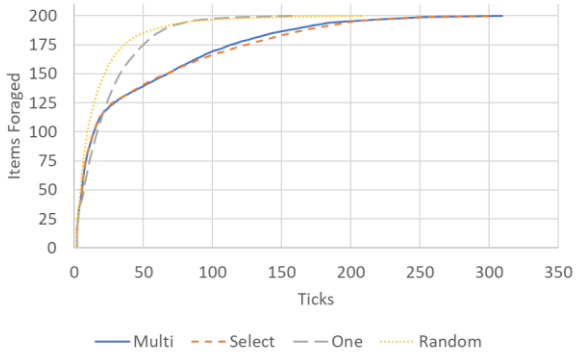
In Figure 6 (b), the Robot Imbalance scenario shows only a slight favouring of Random and One Responder strategies until most items are gathered, but the imbalance of robots then leads to both strategies taking much longer to complete the task than the other approaches. Again, strategy



(a) Equal



(b) Robot Imbalance



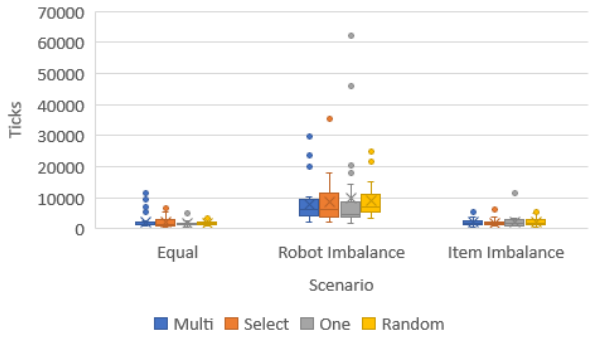
(c) Item Imbalance

Figure 6. Items foraged over simulation ticks for each of the strategies in a 30x30 map: (a) Equal scenario, (b) Robot imbalance scenario, (c) Item imbalance scenario.

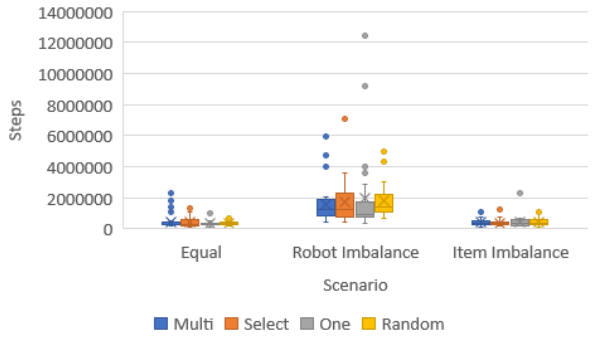
selection during the task could recognise this situation and adopt the strategy most suited.

If individual robot failure is considered, a robot imbalance can occur during the task. A system that can monitor the current swarm composition as well as estimate the progress in the task would therefore be able to adopt a suitable strategy in response to such unpredictable change.

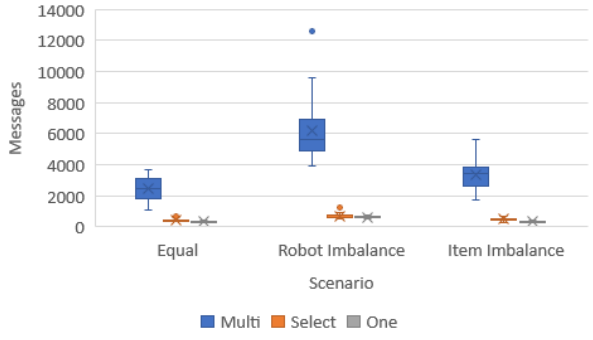
Figure 7 shows the ticks to complete, steps taken, and messages sent for the cooperation strategies in the larger 90x90 grid. Here, it can be seen that the performance of each strategy tends towards that of no cooperation, with large variances in the data and, other than the number of messages sent, similar average values for each metric in each scenario.



(a) Simulation ticks

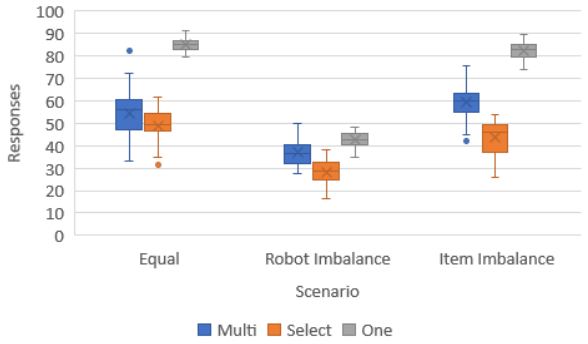


(b) Steps taken

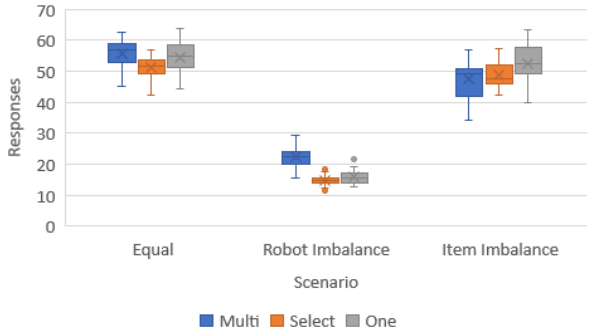


(c) Messages sent

Figure 7. Metrics for each cooperation strategy in a 90x90 map. Circles represent data outliers: (a) simulation ticks, (b) total steps taken, (c) messages sent



(a) 30x30 map



(b) 90x90 map

Figure 8. Percentage of help requests receiving at least one response, for each cooperation strategy and scenario. Circles represent data outliers: (a) 30x30 map, (b) 90x90 map.

It may be expected that the larger map explains the results as messages are not being broadcast far enough in order be received, but a comparison of data in Figure 8 shows that this is not necessarily the case. The proportion of requests receiving a response does not change much between the map sizes for the Multiple and Selective Responders cases, other than when there is a robot imbalance where it can be understood the chances of a robot of the correct type being nearby is significantly lower in a larger area.

The One Responder strategy can be seen to have a much higher percentage of requests receiving a response than the other approaches in a 30x30 map. This is due to the other approaches causing robots who would be able to help to be otherwise engaged in moving to forage an item, and thus unable to respond until they complete that help task. As the One Responder strategy causes only one robot at most to take on a task, other robots remain to be selected. In the 90x90 map, this then drops because of the distance between robots, and more closely matches the other approaches.

The dominant effect in the 90x90 map is the random exploration of the environment, and can be seen in the time taken to complete the task and understood by considering that the number of items remains the same between the two maps. As such, only 2.5% of the cells in a 90x90 map have

an item, compared to 22.2% of the cells in the 30x30 map. It is this decreased chance of stumbling upon an item that has the strongest effect on swarm performance.

The results suggest that allowing a swarm to adjust its cooperation strategy during a task, rather than relying on an initial strategy, could prove beneficial to performance by allowing the swarm to adjust its approach in response to the situation.

B. *Extended Research*

With 600 scenarios and 85 strategy variants, a direct comparison similar to the above is not possible with the extended data. However, as the aim is to show that there is no dominant strategy, a broader overview is sufficient.

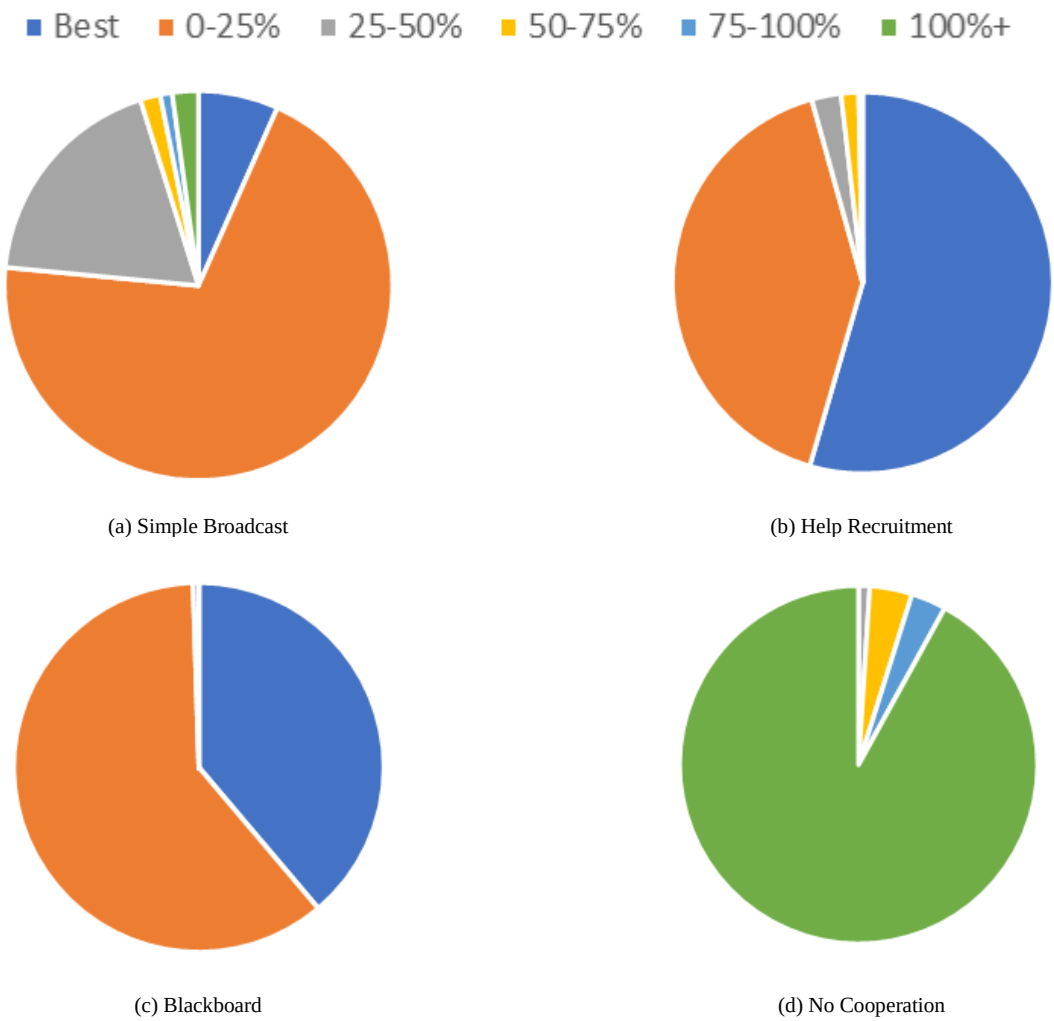
Figure 9 is a summary of the performance of each strategy across all 600 scenarios, as determined by the best performing strategy variant in each scenario, relative to the best performance from all four strategies.

The Help Recruitment and Blackboard strategies perform strongest, in that they each have a high proportion of scenarios for which they are the best performing strategy. At the other end of the scale, the No Cooperation strategy performs poorest, in line with expectations, usually taking more than twice as long to complete as the best performer in a given scenario.

This suggests that the initial hypothesis is confirmed and that no dominant strategy exists, as a scenario best suited to each of the three cooperative strategies may be found. Further, there is no set of parameters for any one strategy that performs well in all scenarios.

Examining the Blackboard strategy further, we see that in all but three of the cases, it performs within 25% of the best performing strategy, and in fact its worst relative performance sees it take just 28% longer than the best strategy, whereas both Simple and Help Recruitment may perform worse in certain scenarios, each having at least one scenario where they take more than twice as long as the best strategy.

Figure 10 compares each strategy according to the



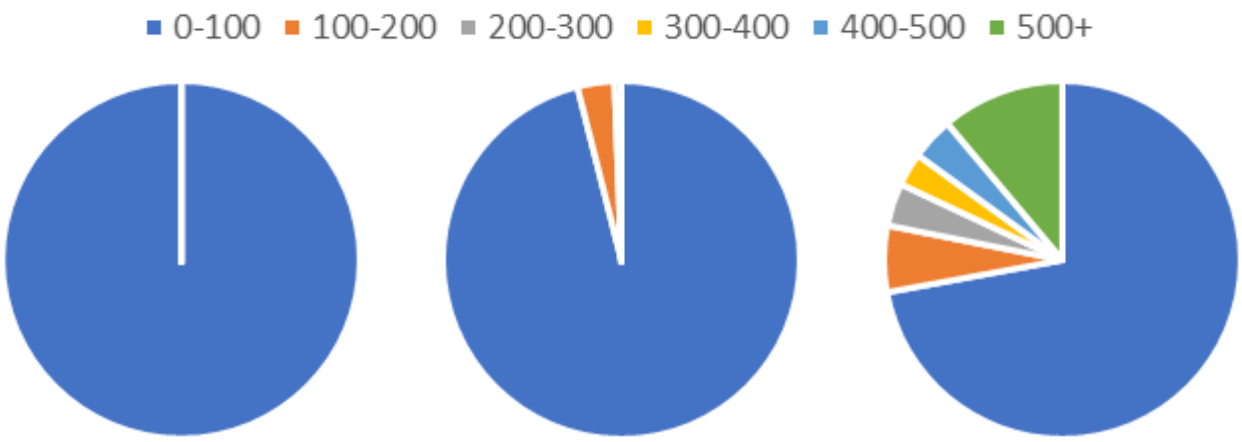
(a) Simple Broadcast

(b) Help Recruitment

(c) Blackboard

(d) No Cooperation

Figure 9. Performance of each strategy across all scenarios. Percentages indicate additional time to complete task relative to best performing strategy.



(a) Simple Broadcast (b) Help Recruitment (c) Blackboard

Figure 10. Performance advantage of each strategy, for scenarios where it is the best performer. Values indicate the difference in simulation ticks to complete the task between the strategy, and the second best option.

performance advantage of each strategy for the scenarios where it is the best performer. The Simple Broadcast strategy offers only a small performance benefit, averaging 6.9 ticks better than the nearest challenger, and with a maximum recorded advantage of 22.5 ticks. Help Recruitment fares little better, averaging 26.8 ticks with a maximum of 208. The Blackboard strategy, on the other hand, shows a clear advantage in many scenarios with an average boost of 264.0 ticks, and a maximum recorded lead of 3515.

This would suggest that the Blackboard strategy, while not dominant in terms of being the best strategy in any given scenario, performs strongly enough in each case that it could be considered de facto dominant, with minimal benefit offered in choosing another strategy and the associated complexity of implementing an autonomous system to make that choice.

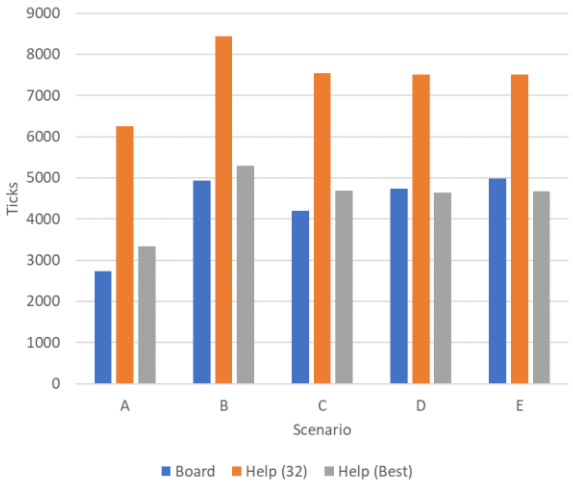


Figure 11. Simulation ticks to complete the five scenarios with the largest performance difference between the Blackboard and Help Recruitment strategies, before and after extending the maximum broadcast range.

However, a closer look at the scenarios in which the Blackboard strategy is dominant reveals a different story. There are a total of 19 scenarios in which the Blackboard outperforms its nearest rival by over 1000 ticks, and each involves the largest map, with the smallest amount of robots with a ratio of 1 white to 7 red robots. As a result, the density of the robot swarm, and in particular the density of the smallest group of robots that would expect to be the target of most requests for help, is at its lowest.

The maximum broadcast range used for the Help Recruitment strategy in this work was 32 cells. Such a range covers less than 20% of the map, targeting one of just eight robots that may be found elsewhere, so it is not surprising that the strategy struggles when the swarm density is reduced. Simulating the 128x128 scenarios for Help Recruitment using increased ranges of 48, 64, 80 and 96 cells produced comparable results to the Blackboard strategy, as seen in Figure 11.

Rather than implementing a complex system to select a strategy based on identifying the scenario a swarm finds itself in, the results indicate that either a Help Recruitment or a Blackboard strategy would provide suitable performance given the right parameters. An autonomous system may still be used in order to adjust those parameters accordingly.

In addition, it may be better to use the simpler of the strategies with the least overhead – a Help Recruitment strategy requires no memory of items found and foraged, and does not require any periodic communication with the other members of the swarm. On the other hand, energy considerations may make large broadcast ranges unsuitable, and the short range communications between robots in the Blackboard strategy may be more efficient. Further research on this will be needed to investigate the relative impact of each strategy.

VIII. Results – Simulator Comparisons

The following subsections present the results of the initial research, followed by the extended work which followed.

A. Initial Research

Table VII compares the time taken to complete the task for each of the simulators in the Equal scenario. The time-stepped simulation presented here is significantly faster than the original multi-threaded simulation. This can largely be accounted for by the deliberate delays introduced previously to allow for visualization, with some impact of the reliance on real-time delays for communication, which makes a true comparison difficult. The table also shows the execution time for the One Responder protocol as implemented in the updated multi-threaded simulator, showing it is still slower than the time-stepped approach.

Figure 12 shows that the time-stepped simulation takes a much larger number of steps in the Multiple and Selective Responder strategies, and also shows an increase under One Responder. This unexpected result may be explained by the specific behaviour of the robots in each simulation. In the multi-threaded approach, robots pause frequently, the effect of which is that fewer robots will move in each step. For example, on deciding to respond to a help request a robot pauses for three seconds. Further, if another help request is received during that pause, that too may be processed and the robot may choose to act on that, with a further pause.

The effect of these pauses is to reduce the number of robots moving at any given time. In the time-stepped simulation, a robot will only pause when waiting for help responses or assignments in the One Responder strategy.

It is notable that despite these pauses, robots in the multi-threaded approach take fewer steps overall, rather than taking the same number of steps over a longer period. This suggests there may be a benefit to reducing the number of robots engaged in random exploration, but this will need to be investigated further.

The impact of the two simulations on the host platform was compared and Table VIII displays the approximate processor and memory usage of the two platforms when running simulations.

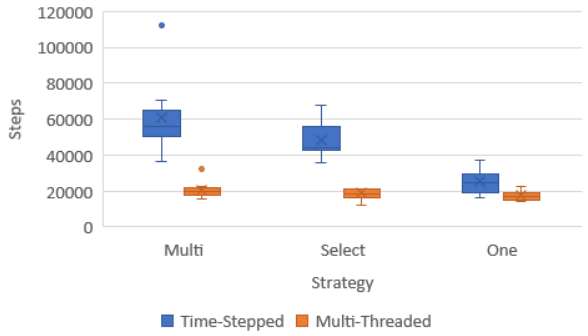


Figure 12. Total steps for each simulator using the three strategies in the Equal scenario. Circles represent data outliers.

TABLE VII. Simulation Duration (Equal Scenario)

Simulator	Time (s)		
	Multiple Responders	***Selective Responders***	***One Responder***
Time-Stepped	2.11	0.89	0.17
Multi-Threaded v1	155.14	142.01	130.86
Multi-Threaded v2			1.35

TABLE VIII. Simulator CPU and Memory Usage

Simulator	CPU Usage (%)	Memory Usage (MB)
Time-Stepped (Display)	5-6	30-35
Time-Stepped (No Display)	55-60	30-35
Multi-Threaded	35-40	700-750

Overall, the time-stepped approach will put less strain on the CPU, as despite its higher usage during execution without a display, it will run for a fraction of the time. With a display, the execution is halted between ticks to update the display at a framerate of the user's choosing, and so CPU usage drops. The multi-threaded simulation has no option to disable display updating, but the use of a separate thread for each robot results in a moderate level of CPU usage for a longer period of time.

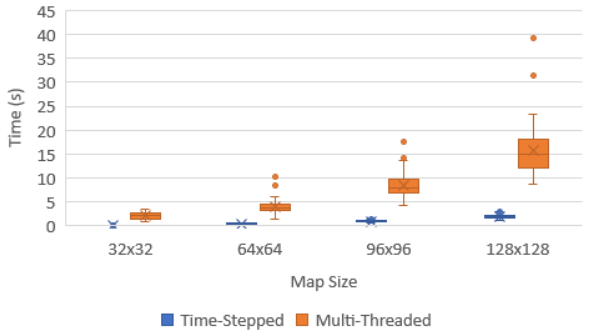
The lower memory footprint of the time-stepped simulation is most likely due to specific implementation differences. Each robot in the multi-threaded simulation contains a copy of the world map and lists of robots and items, whereas the time-stepped simulation uses a shared resource. While requiring local copies is a factor in any real scenario, it is not required to simulate that unless it is expected that robots will have different local data. If this is a requirement, the memory usage would increase accordingly.

B. Extended Research

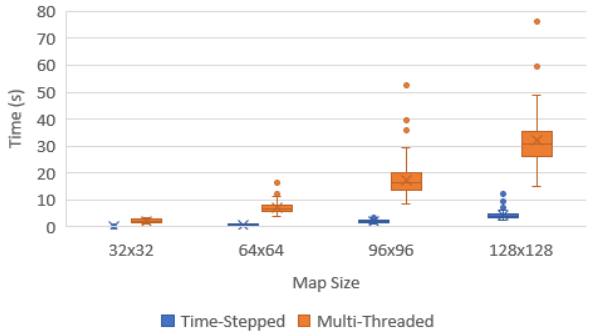
The implementation differences between the time-stepped and multi-threaded simulator used in the initial research have made a performance comparison difficult to achieve, prompting the rewrite of the multi-threaded simulator to use identical robot behaviour. The following results are based on comparisons run between the time-stepped simulator and the re-written multi-threaded simulator.

Figure 13 shows the mean time taken for each simulator to complete the task on each map size, for the Help Recruitment and No Cooperation strategies respectively. It is clear from the results that the multi-threaded simulator is slower at completing each simulation, and it can also be seen that the duration increases more quickly with an increase in map size for the multi-threaded implementation.

Figures 14 and 15 show the simulation ticks and total steps taken within the simulator for each implementation and map size. As discussed in Section V, the multi-threaded simulator uses the mean number of updates executed by the robots as an estimate of the ticks taken by the simulation to complete. The results here show that in both simulators, the

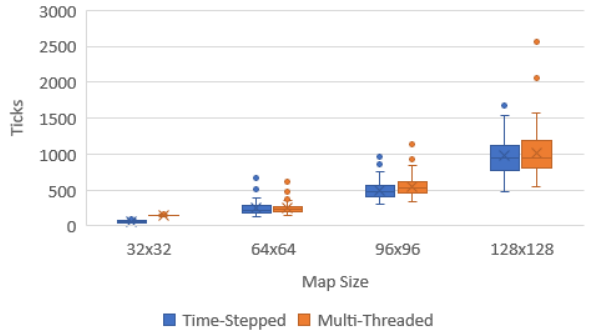


(a) Help Recruitment

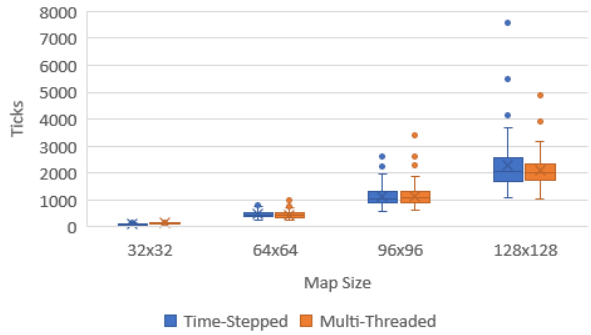


(b) No Cooperation

Figure 13. Average simulation execution time across all four maps. Circles represent data outliers: (a) Help Recruitment strategy, (b) No cooperation strategy.



(a) Help Recruitment



(b) No Cooperation

Figure 14. Average simulation ticks (time-stepped) or robot updates (multi-threaded) across all four maps. Circles represent data outliers: (a) Help Recruitment strategy, (b) No cooperation strategy.

use of the Help Recruitment strategy is able to reduce the number of ticks, and by extension the number of steps taken by the swarm.

It can also be seen that the results are broadly similar between the two simulations, except for the 32x32 map, where the multi-threaded implementation takes more ticks on average, and more steps, during the Help Recruitment strategy. A further investigation into the behaviour of the multi-threaded simulator shows that the robots each receive a different number of updates, as might be expected, but also that some robots do not receive any updates at all throughout the course of the simulation.

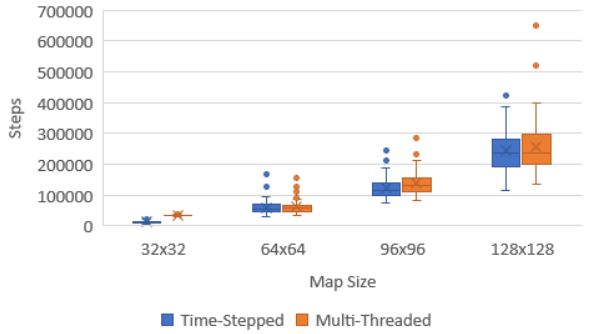
Figure 16 shows the average number of help offers sent in response to each help request when using the Help Recruitment strategy on each map size. It can be seen that the multi-threaded approach results in fewer offers generated per request. This may be due to the fact that robots are not always given processing time, however as the messages are buffered, a response will be sent provided the robot is updated at least once since receiving the request.

The average number of help requests sent on each map size is seen in Figure 17, where it can be seen that the multi-threaded simulator sends more requests than in the time-stepped approach. Again, this may be because of the processing time imbalance. If fewer robots are able to respond to a request, there remain more items for the robots to encounter, and thus more occasions where help requests will be sent.

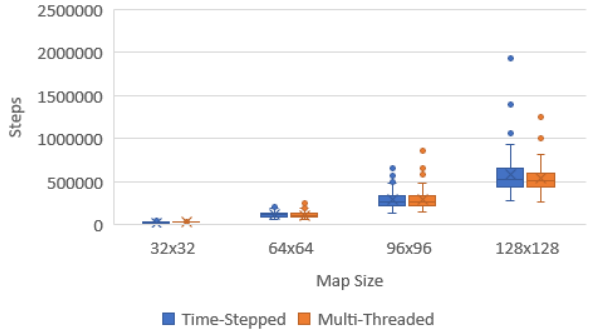
Investigations into the memory and CPU usage of the two simulators showed no appreciable difference, suggesting that the differences shown for the previous multi-threaded approach were down to implementation differences rather than inherent in the time-stepped or multi-threaded approach chosen.

Overall, these results show that despite the underlying implementation of the robot behaviour being the same in each simulator, the scheduling approach taken by each can result in different behaviour. As the simulation duration increases, these effects have less of an outcome on the final result.

Delegating the execution of the robots to the CPU scheduler can result in unpredictable behaviour by the robots, and results in a reduced degree of control in the simulation. A time-stepped approach ensures that each robot receives an identical amount of execution time in a predictable way. Furthermore, the time-stepped approach is significantly faster than the multi-threaded simulator, allowing more simulations to be run in a shorter period of time.



(a) Help Recruitment



(b) No Cooperation

Figure 15. Average steps taken by all robots, across all four maps. Circles represent data outliers: (a) Help Recruitment strategy, (b) No cooperation strategy.

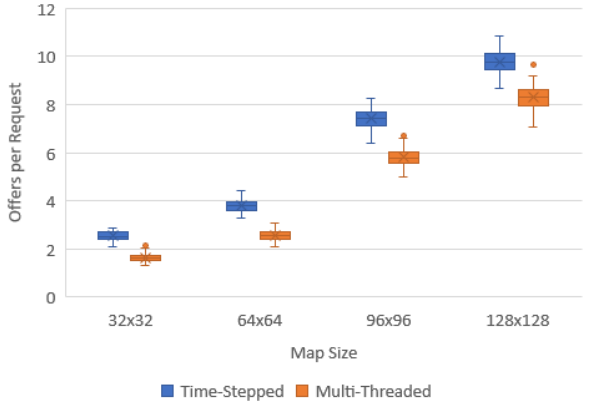


Figure 16. Help offers per request sent, across all four map sizes. Circles represent data outliers.

IX. Conclusion and Future Work

The presented research used a time-stepped simulation to investigate the effects of different cooperation strategies for a swarm carrying out a foraging task. It was shown that the performance of a strategy is sensitive to the scenario in which the swarm operates, and so sticking with a single strategy may lead to suboptimal performance. However, while the initial findings leaned towards some potential for changing the underlying strategy mid-mission, the extended research presented here suggests similar performance gains

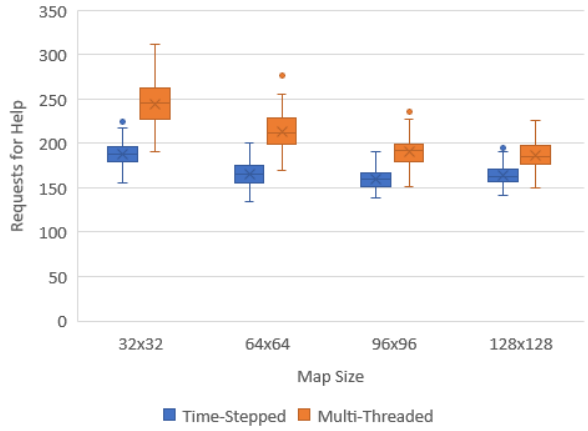


Figure 17. Help requests sent, across all four map sizes. Circles represent data outliers.

can be obtained simply by adjusting the parameters of an existing strategy.

Further, different stages of the task appear to favour different approaches. During the initial phase where large numbers of items remain to be discovered, random exploration with no cooperation strategy produces the best results. Only when a small proportion of the items remain does the adoption of a cooperation strategy start to benefit the performance of a swarm.

There is therefore a potential benefit to be gained by implementing a self-adaptive system that can modify these parameters based on the swarm's understanding of the current scenario, allowing for faster completion of the task. If the factors that affect selection of suitable parameters are simple, such as using swarm density alone to determine a broadcast range, then this may be achieved using a simple calculation. However, if the factors are sufficiently numerous and complex, there may still be a benefit to using simulation within the swarm to explore the possibility space without the risks associated with executing inefficient behaviours in reality.

These results are based on a simplified simulation of the foraging task. As noted in Section II, simulation can be a useful means of testing complex scenarios in situations where making use of large numbers of robots may be considered infeasible. Nonetheless, corroboration of the data with results from other simulations, and possibly physical robotic tests, would be useful.

The time-stepped simulation was compared against a real-time, multi-threaded approach and found to execute faster and with more reliable results, thanks to each robot being guaranteed an equal amount of processing time throughout the task. This would make the time-stepped simulation more suitable for use as part of the MAPE-K loop for a foraging swarm, forming MAPSE-K (Figure 18) [2]. This could be achieved by embedding the simulator on one or multiple robots within the swarm, to analyse and adjust the strategy without risking reduced performance.

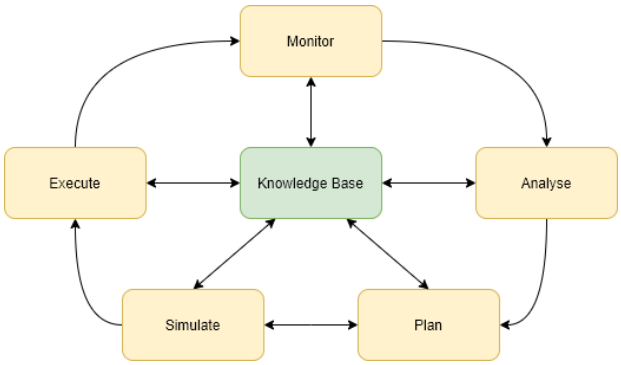


Figure 18. MAPSE-K loop. Simulation is used to test the plans before executing the best performing option, or returning to the planning stage.

The expected limited processing capabilities of the host robot mean managing the overhead that simulation entails will become a major factor. A time-stepped approach allows execution to be stopped and restarted with predictable results, as well as executing quicker overall, so can be less demanding on the host platform.

Future work will explore the methods of giving the swarm an autonomic ability to adjust the Help Recruitment broadcast range in response to the scenario faced. Other factors that can affect the performance will also be investigated, such as the impact of energy costs, as well as changes to the scenarios that may prompt the swarm to adjust their strategy accordingly.

REFERENCES

[1] L. McGuigan, C. Saunders, R. Sterritt, and G. Wilkie, 'Cooperation Strategies in a Time-Stepped Simulation of Foraging Robots', in *The Twelfth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2020) IARIA*, Oct. 2020, pp. 135-142.

[2] R. Sterritt et al., 'Inspiration for Space 2.0 from Autonomic-ANTS (Autonomous NanoTechnology Swarms) Concept missions', presented at the Reinventing Space Conference, 2019.

[3] A. Farahani, G. Cabri, and E. Nazemi, 'Self-* properties in collective adaptive systems', in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, Heidelberg, Germany, Sep. 2016, pp. 1309–1314.

[4] L. Bayindir, 'A review of swarm robotics tasks', *Neurocomputing*, vol. 172, pp. 292–321, Jan. 2016.

[5] I. Navarro and F. Matía, 'An Introduction to Swarm Robotics', *ISRN Robot.*, vol. 2013, pp. 1–10, 2013.

[6] G. Beni, 'From Swarm Intelligence to Swarm Robotics', in *Swarm Robotics*, Berlin, Heidelberg, 2005, pp. 1–9.

[7] IBM, 'An architectural blueprint for autonomic computing, 4th ed.' IBM White Paper, 2006.

[8] J. O. Kephart and D. M. Chess, 'The vision of autonomic computing', *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[9] R. Sterritt, G. Wilkie, G. Brady, C. Saunders, and M. Doran, 'Autonomic robotics for future space missions', 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2015) - ESA/ESTEC, Noordwijk, Netherlands, Sept. 2015.

[10] M. Doran, R. Sterritt, and G. Wilkie, 'Autonomic architecture for fault handling in mobile robots', *Innov. Syst. Softw. Eng.*, Apr. 2020, doi:10.1007/s11334-020-00361-8

[11] M. Torres-Torriti, T. Arredondo, and P. Castillo-Pizarro, 'Survey and comparative study of free simulation software for mobile robots', *Robotica*, vol. 34, no. 4, pp. 791–822, Apr. 2016.

[12] J. Prasetyo, G. De Masi, and E. Ferrante, 'Collective decision making in dynamic environments', *Swarm Intell.*, vol. 13, no. 3, pp. 217–243, Dec. 2019.

[13] J. Zelenka, T. Kasanický, and I. Budinská, 'A Self-adapting Method for 3D Environment Exploration Inspired by Swarm Behaviour', in *Advances in Service and Industrial Robotics*, Cham, 2018, pp. 493–502.

[14] N. Capodieci, E. Hart, and G. Cabri, 'An Artificial Immunology Inspired Approach to Achieving Self-Expression in Collective Adaptive Systems', *ACM Trans. Auton. Adapt. Syst. TAAS*, vol. 11, no. 2, p. 6:1-6:25, Jun. 2016.

[15] N. Bredeche, E. Haasdijk, and A. Prieto, 'Embodied evolution in collective robotics: A review', *Front. Robot. AI*, vol. 5, p. 12, 2018.

[16] K. S. Kappel, T. M. Cabreira, J. L. Marins, L. B. de Brisolara, and P. R. Ferreira, 'Strategies for Patrolling Missions with Multiple UAVs', *J. Intell. Robot. Syst.*, vol. 99, pp. 499–515, Sep. 2019.

[17] G. Leu and J. Tang, 'Survivable Networks via UAV Swarms Guided by Decentralized Real-Time Evolutionary Computation', in *2019 IEEE Congress on Evolutionary Computation (CEC)*, Jun. 2019, pp. 1945–1952.

[18] M. Frasheri, B. Cürüklü, M. Esktröm, and A. V. Papadopoulos, 'Adaptive Autonomy in a Search and Rescue Scenario', in *2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, Sep. 2018, pp. 150–155.

[19] F. Yan, K. Di, J. Jiang, Y. Jiang, and H. Fan, 'Efficient decision-making for multiagent target searching and occupancy in an unknown environment', *Robot. Auton. Syst.*, vol. 114, pp. 41–56, Apr. 2019.

[20] C. Saunders, R. Sterritt, and G. Wilkie, 'Collective Communication Strategies for Space Exploration', *J. Br. Interplanet. Soc.*, vol. 72, no. 12, pp. 416–430, 2019.

[21] M. Puviani, G. Cabri, and L. Leonardi, 'Enabling Self-Expression: The Use of Roles to Dynamically Change Adaptation Patterns', in *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, Imperial College, London, United Kingdom, Sep. 2014, pp. 14–19.

[22] M. Dorigo, G. Theraulaz, and V. Trianni, 'Reflections on the future of swarm robotics', *Sci. Robot.*, vol. 5, no. 49, eabe4385, Dec. 2020.

[23] C. Pinciroli *et al.*, 'ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems', *Swarm Intell.*, vol. 6, no. 4, pp. 271–295, Dec. 2012.

[24] N. Jakobi, P. Husbands, and I. Harvey, 'Noise and the reality gap: The use of simulation in evolutionary robotics', in *Advances in Artificial Life*, vol. 929, F. Morán, A.

Moreno, J. J. Merelo, and P. Chacón, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 704–720.

[25] F. Kamrani and R. Ayani, 'Using On-line Simulation for Adaptive Path Planning of UAVs', in *11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'07)*, Oct. 2007, pp. 167–174.

[26] N. Keivan and G. Sibley, 'Realtime simulation-in-the-loop control for agile ground vehicles', *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 8069 LNAI, pp. 276–287, 2014.

[27] C. Pepper, S. Balakirsky, and C. Scrapper, 'Robot simulation physics validation', in *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, Washington, D.C., Aug. 2007, pp. 97–104.

[28] C. J. E. Castle, N. P. Waterson, E. Pellissier, and S. Le Bail, 'A Comparison of Grid-based and Continuous Space Pedestrian Modelling Software: Analysis of Two UK Train Stations', in *Pedestrian and Evacuation Dynamics*, Boston, MA, 2011, pp. 433–446.