Task partitioning for foraging robot swarms based on penalty and reward

Edgar Buchanan Berumen

Ph.D.

University of York
Electronic Engineering

November, 2018

Abstract

This thesis is concerned with foraging robots that are retrieving items to a destination using odometry for navigation in enclosed environments, and their susceptibility to dead-reckoning noise. Such noise causes the location of targets recorded by the robots to appear to change over time, thus reducing the ability of the robots to return to the same location. Previous work on task partitioning was attempted in an effort to decrease this error and increase the rate of item collection by making the robots travel shorter distances.

Dynamic Partitioning Strategy (DPS) is introduced and explored in this thesis which adjusts the travelling distance from the items location to a collection point as the robots locate the items, through the use of a penalty and reward mechanism. Robots adapt according to their dead-reckoning error rates, where the probability of finding items is related to the ratio between the penalty and the reward parameters.

In addition, the diversity in the degrees of error within the members of a robot swarm and the performance repercussion in task partitioning foraging tasks is explored. This is achieved by following an experimental framework composed of three stages: emulation, simulation and hardware. An emulation is generated from an ensemble of machine learning techniques. The emulator allows to perform enriched analyses of simulations of the swarm from a global perspective in a relatively low time compared with experiments in simulations and hardware. Experiments with simulation and hardware provide the contribution of each robot in the swarm to the task.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deepest gratitude to my supervisors Jon Timmis and Andy Pomfret for providing me with their insightful guidance, yet at the same time giving me so much freedom to explore different research directions.

I will be forever grateful to my Mum (Malena) and my Brother (Aldo), my family for their constant support. In addition, I would like to thank as well my father (Silvestre) wherever he is for all the values and wisdom that he taught me as a kid. Of course to my favourite two dogs: Rosita and Pupy.

My sincere thanks to all my friends that have been there for me through this Ph.D. adventure. Bernardo, Gerardo, Perla, Walid, Fabiola and Ieva.

Last but not least, I would like to thank the Mexican National Council of Science and Technology (CONACyT) and the University of York, for giving me the opportunity to pursue part of my dream.

Declaration

I declare that the research described in this thesis is original work, which I undertook at the University of York during 2014 - 2018. This work has not previously been presented for an award at this, or any other, University.

Some parts of this thesis have been published in conference proceedings; where items were published jointly with collaborators, the author of this thesis is responsible for the material presented here. For each published item the primary author is the first listed author.

Conference Papers

E. Buchanan, A. Pomfret, and J. Timmis, "Dynamic Task Partitioning for Foraging Robot Swarms." in *International Conference on Swarm Intelligence*, 2016.

Chapter 1

Introduction

1.1 Overview

The foraging task, inspired from biology such as the behaviour of ants and bees, has been widely studied in the field of swarm robotics (Bayındır, 2016). This task consists of robots exploring an environment searching for items to transport to a home area. However, until recently, little research has been done in terms of fault-tolerance in foraging robotic swarms.

The potential applications range from agriculture and transportation of delicate materials to asteroid mining. In case of agriculture where the population keeps increasing, food has become a key element for humanity survival. Therefore, the production of food needs to increase and with swarm robotics it is possible to achieve this due to the high redundancy of robots. Also, if robots experiences error, robots are able to compensate the error with the strategy proposed in this thesis and food production will continue without any halts.

In case of a nuclear disaster or oil spill a group of robot could be deployed to collect the oil or any nuclear waste without any human intervention. Even the robots were faulty and human intervention to repair them would not be possible because the environment might be too dangerous, the robots would continue the collection of these delicate materials.

In the future robots implementing the strategy proposed in this thesis could be deployed to take medicine directly to the affected area inside of people. In case of asteroid mining robot would collect to the home base the material found in the asteroid in a safe way.

Robots using dead-reckoning as navigation are susceptible to noise due to systematic and non-systematic errors (Carlson and Murphy, 2005). Although that former can be relatively easy to compensate (Borenstein and Feng, 1996), the latter is more complicated to compensate due to its non-linear nature. This error causes inaccurate navigation and, hereby, robots are unable to retrieve items from a previously recorded landmark.

Early work addressed this issue by dividing the main foraging task into multiple smaller fixed subtasks (Pini et al., 2014), such that the robots would travel shorter distances and the error would decrease due to the lack of noise accumulation over the distance travelled. However, this approach is not robust. The number of subtasks defined by the user would not provide the same expected results if a different robot platform is used, the error in the robot changes or the environment changes.

Later work increased the robustness of the task partitioning by making it autonomous (Pini et al., 2013). The robot would define the size of the partitioning according to its performance in the subtask. However, when using this approach, the robots take a long time to learn the appropriate partition size.

The aim of this thesis is to present a strategy to increase the performance of the robots experiencing dead-reckoning noise. This strategy would allow the robot find the appropriate partition size according to a penalty and reward mechanism. The partition would be different for each robot according to its performance.

1.2 Hypothesis

The following hypothesis is proposed to guide the work presented in this thesis:

"*A penalty and reward learning mechanism allows the robots to learn the best partition size that allows them to divide the foraging task into multiple smaller sub-tasks, and by this, decrease the accumulated error. The partition would adapt and be individualised for each robot according to its own performance.*"

Adaptability is important for robots that operate for long periods of time. As discussed, errors of the robots can increase over time, or the operational environment may change. These aspects of adaptability are the focus of this thesis.

1.3 Thesis Contributions

This thesis makes a number of substantial contributions to the research in dynamic partitioning in foraging and error diversity within robotic swarms. The most significant contributions are summarised below.

1. This thesis presents the Dynamic Partitioning Strategy (DPS), which is a penalty and reward mechanism that a allows the robots to divide the foraging task into smaller components. DPS is shown to be capable to increase the item collection accuracy in robots. Additionally, when applied to different degrees of error, each robot learns a different partition length according to its degree of error.

2. This thesis rigorously and extensively explores the discrepancy between the results when comparing diversity and uniformity error across the swarm from a global perspective.

This is achieved by using different statistical techniques to analyse a set of emulations. It is important to consider heterogeneous and homogeneous errors when performing sensitivity analyses. On one hand, with homogeneous error it is possible to understand the relation between each parameter and each output. On the other hand, with heterogeneous error it is possible to identify the different degrees of robustness in the swarm in terms of the number of robot experiencing extensive error.

3. This thesis describes how each robot contributes in a different way to the item collection when there is error diversity across the swarm. This is done by analysing the different degrees of flexibility with each strategy in terms of the swarm size, distance between items source and home area and degree of error.

4. Finally, this thesis concludes that the number of robots experiencing extensive error defines the performance of the task. This is achieved by measuring the social entropy for the swarm with heterogeneous error.

1.4 Thesis Outline

This section outlines the remaining chapters of this thesis.

Chapter 2 provides a detailed literature review of chain formations in foraging algorithms. Firstly, the importance of fault-tolerance in robotic swarms systems is highlighted. Then, a foraging robot swarms summary is presented. Lastly, task partitioning algorithms are described.

Chapter 3 presents the methodology used in the experiments presented in this thesis. Robotics platforms and robot simulator are firstly described. Then, parameters and outputs are introduced with their respective metrics. Lastly, statistical tests and techniques used to analyse and understand the experiments are described in detail.

Chapter 4 describes the different properties of the novel task partitioning strategy introduced in this thesis: Dynamic Partitioning Strategy. Convergence to solution, comparison between strategies and effect of changing the ratio between the penalty and rewards are some of the experiments in simulation documented in this chapter.

Chapter 5 critically analyses each partitioning strategy with heterogeneous and homogeneous error with emulations. Interactions between the parameters and the outputs for each strategy are studied. In addition, results from statistical techniques highlight the contrast of considering error diversity and error uniformity.

Chapter 6 explores in more detail in simulations and hardware how each robot in the swarm reacts and contributes to the foraging task for error diversity and uniformity across the swarm. This is achieved by measuring the performance of each robot.

Chapter 7 presents the conclusions of this thesis and suggests some ideas for further research.

Chapter 2

Literature Review

2.1 Introduction

Swarm robotics is defined as a decentralized group of robots interacting between themselves and the environment. They are characterized because of their emergent properties rising from simple interactions and by the way that their ability take their own decisions. Swarm robotics overcomes problems that are common in centralized systems such as lack of scalability and flexibility. In addition, they are robust because of their high redundancy in robots. In other words, in robotic swarms there is no a single point of failure (Barca and Sekercioglu, 2012). However, recent literature has shown that swarm robotics is not inherently robust for all scenarios as it was initially thought. Specific partial failures can lead the functionality of the swarm to be deteriorated (Winfield and Nembrini, 2006; Pini et al., 2014).

The task studied in this thesis is foraging which consists of robots retrieving items from the environment to a home area. Partial failures in the robots produce dead-reckoning error that affects the accuracy of their navigation. The performance degrades due to landmarks recorded by the robots to be inaccurate. Pini et al. (2014) proposed to decomposed the main foraging task into multiple smaller subtask. Items are collected from the source to the home area by a series of item transactions between robots. Since the robots are travelling shorter lengths, the error accumulation would be smaller. More details about task partitioning can be found in Section 2.3.

In this thesis, Dynamic Partitioning Strategy is introduced in which the robots divide the task in an emergent way by using a penalty and reward mechanism. The partition length for each robot increases as items are found and decreases as items are lost.

2.2 Swarm Robotics

Swarm robotics is defined by Sahin (2005) as:

"*Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment.*"

Swarm robotics inherits characteristics from the field of robotics such as autonomy. In other words, robots have the capability of performing tasks without any human intervention. The second characteristic is that robots interact with their environment by using their actuators. Finally, robots are aware of their surroundings by using sensors and local communication Brambilla et al. (2013).

Swarm robotics also inherits three properties from the swarm intelligence field: scalability, flexibility and robustness (Brambilla et al., 2013). The first property, scalability, allows the addition and removal of robots in the swarm without affecting its functionality. Flexibility is the second property where the swarm is able to adapt to the environment where it is working. Lastly, robustness is the property in which if a robot is damaged, functionality is not perturbed due to the high redundancy in the number of robots as long as the swarm size is above a minimum number of robots (Bjerknes and Winfield, 2012; Lau et al., 2013). However, robustness is not always achieved and this is described next.

2.2.1 Why is fault-tolerance important in Swarm Robotics?

Authors in (Winfield et al., 2005; Winfield and Nembrini, 2006) explored the interactions and repercussions of robots with faults in a swarm containment task (Figure 2.1). In the swarm containment task a group of robots performs swarm taxis, a variation of aggregation where the robots feel attracted to a landmark. Robots broadcast their own ID and their neighbours' IDs. A robot considers itself lost when the number of neighbours surrounding it is below a predefined threshold value. If the robot is lost then it turns 180 degrees and performs an obstacle avoidance behaviour. The task is achieved when the robots surround the target.

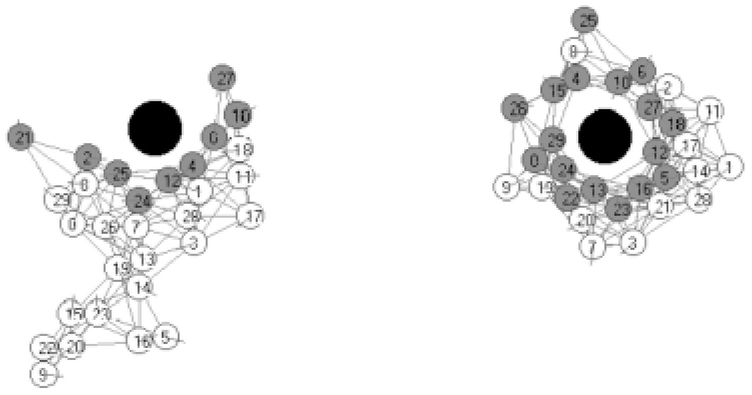


Figure 2.1: Emergent encapsulation: (left) encapsulation in progress and (right) encapsulation complete Winfield and Nembrini (2006) .

All potential robot faults were classified and evaluated according to their impact on the swarm performance and functionality by using Failure Mode and Effect Analysis (FMEA). The robot faults classification are: *motor*, *communications*, *obstacle avoidance sensors*, *beacon sensor*, *control system* and *total system* failures. The impact of each fault is described next.

A *motor failure* occurs when a single or both motors experience an abnormal behaviour. It is important to mention that the robot remains connected to the network by communicating with other robots. This means that faulty and non-faulty robots interact with each other through messaging. Non-faulty robots feel attracted to the faulty-robots. This is a serious hazard because it can lead non-faulty robots getting trapped with the faulty robots and the containment emergent behaviour might not be achieved.

When a robot is unable to broadcast or receive messages a *communications failure* has occurred. As consequence, the robot is not able to follow the swarm and it gets lost. Non-faulty robots consider faulty robots as movable obstacles. This is not a serious hazard that affects functionality in the swarm. The swarm performance only degrades if the faulty robot is moving within the swarm.

Obstacle avoidance failure occurs when a robot is unable to detect any obstacles in their trajectory. This anomaly does not have serious impact in the swarm because non-faulty robots would try to avoid the faulty robots. Robots might get damaged if two or more robots share the same anomaly causing them to collide into each other. Besides damaging itself and other robots, a faulty robot can also damage the beacon because it is unable to detect its proximity.

Beacon sensor failure occurs when a robot cannot detect the beacon. This failure has little impact in the swarm functionality. Only the performance decreases due to this failure. The main reason is that the faulty robot does not contribute to lead the swarm towards the beacon. The robot can not detect the beacon. Therefore, the faulty robot only follows the general position of the swarm.

Controls systems failure has different impacts depending on the swarm state and in which part of the control system happened. As consequence, if the failure affects the dynamics of the motors, then this a serious fault that potentially could lead to swarm anchoring.

Finally, *total systems failure* occurs when a robot stays stationary and inactive. This can be caused by problem related with power. The robot is not connected to the network, therefore it does not contribute with the swarm movement towards the beacon. Hence, this is the least serious hazard because other robots will consider it as static obstacle and they will avoid it.

As explained before, partial failures in robots have a bigger impact in the swarm than total failures, in particular faults related with motion. In addition, Carlson and Murphy (2005) show that motion failures are common in unmanned ground vehicles. Hence for this reason the authors point out that partial failures should by addressed by implementing mechanisms to counter the anomaly.

2.2.2 Fault-tolerance in Swarm Robotics

In the previous section it was shown that partial failures in a collective group of robots could lead to performance and functionality degradation. As a result, attempts have been made to increase robustness of the swarm in order to either detect (Christensen et al., 2009; Parker, 1996; Tarapore et al., 2015) or, diagnose failures (Li and Parker, 2007; Parker and Kannan, 2006; O'Keeffe et al., 2017) or even to recover from failures (Humza et al., 2009; Long et al., 2003; Timmis et al., 2010).

Fault detection can be achieved by the faulty robot identifying the fault in itself (endogenous detection) or by a non-faulty neighbour (exogenous detection). An example of endogenous detection is the work presented in (Tarapore et al., 2015) where a single robot collects the performance of its neighbours in the form of feature vectors. Then, the robot compares its performance with its neighbours. If a big discrepancy is detected between results, then the robot considers itself as faulty.

An exogenous fault detection approach is presented in (Christensen et al., 2009). A group of light flashing robots synchronize to emit light at the same as fireflies do. A robot is detected as faulty by its neighbours if the robot is out synchronization.

Fault diagnosis is the categorisation of the fault previously detected. In (O'Keeffe et al., 2017), the authors implemented an approach where an external neighbour robot (doctor robot) diagnose faults in faulty robots. This is achieved by using supervised learning where by training the robots, the robots can detect the source of the abnormal behaviour in the faulty robot.

Fault recovery consists in finding a way to repair or prevent any further degradation introduced by a faulty robot. In (Timmis et al., 2010) the authors designed a mechanism that consists in isolating the faulty robot in order to prevent any damage spreading. This achieved by non-faulty robots feeling attracted to the faulty robot.

2.3 Chain formation in foraging

Foraging consists in robots exploring the environment and transporting items to a home area. In this section chain formations in foraging found in literature are described. For this thesis, implementations are classified into three types. *Chain formation*, described in Chapter 2.3.1, where robots are recruited to form a chain. The robots do not change position and remain static in the chain between the items source and the home area. *Virtual pheromone* is used to guide the robots to the items source creating eventually a dynamic chain where robots are flowing between the items source and the home area, shown in Chapter 2.3.2. Lastly, in *task partitioning* the robots decompose the task into smaller subtasks, presented in Chapter 2.3.3.

2.3.1 Chain Formation

Nouyan et al. (2007) proposed robot recruitment mechanism to make chain formations that lead from the home area to the items source. In their work two different mechanisms were presented the first one referred as *chain* that is a a linear formation and the second as *vectorfield* where the formation spreads like a tree shape with branches. A robot change the colour of the LEDs according to its actual state and by using camera other robots will react according to the colour of its neighbours.

Recruitment process for *chain* mechanism works as follows. Firstly, the robots are initially placed in random positions in the environment. The first robot to reach the items source starts the chain. Robots close to the chain have a specific probability of joining the chain tail. The robot in the tail have a probability of leaving the chain. After the chain formation links the items source and the home area, the probability of the robot in the formation tail is reduced to zero. Three variants of chains according to their degrees of motion are studied. In the first one all the robots remain static once join the formation. In the second, robots try to make a straight linear chain. Third, the chain performs circular movement orbiting the home area. Experiments shown that the last two variants perform better because these variants increase the probability of finding the items source with the lowest number of robots.

As mentioned before, the main difference between *vectorfield* and *chain* formation is that *vectorfield* expands with branches. Robots can join any position in the chain and not necessary join in the tail. The downside of *vectorfield* is that it takes more time to form the chain that connects items source and home area. This is because more robots get idle being part of the chain instead of exploring. *Chain* and *vectorfield* formations are shown in Figure 2.2.

The authors showed that their implementation is fault-tolerant. This is shown by disabling specific components on robots to recreate partial failures. When there is no camera, robots become movable obstacles. Robots without LEDs can join the chain, however they do not contribute to the chain formation therefore they are considered as obstacles and eventually faulty robots leave the chain. Lastly, if there is an obstacle avoidance sensor failure, robots can still join the chain but performance is degraded.

In (Groß et al., 2008; Nouyan et al., 2009) a variation is implemented with physical robots and it is expanded by making robots retrieve items to the home area by following the chain. If the weight of the items is too heavy for a single robot, robots cooperate as a team to carry the item to the home area. Robots attached to items change their colour to red to indicate that they are movable obstacle to other robots. This causes other robots to approach and attach to that robot.

In Schmickl et al. (2011) a similar chain implementation is done with an underwater swarm project called CoCoRo. In this project all the robots start exploring from a base station and, in a similar way, robots join and leave the chain with a specific probability. The chain ends once

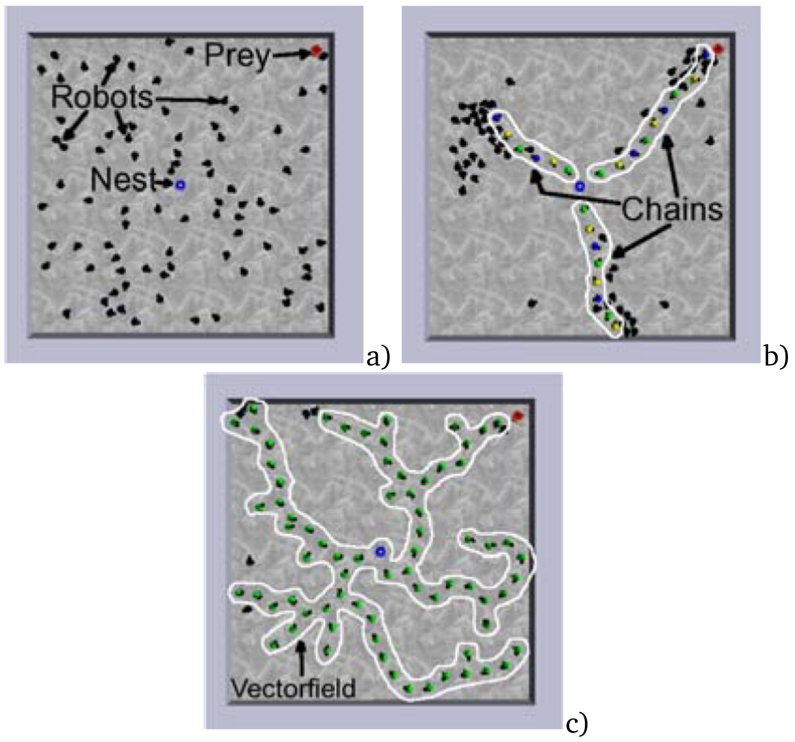


Figure 2.2: Robots exploring the environment (a), making a chain formation (b) and a vectorfield formation (c). Nouyan et al. (2007) .

the target is found.

The main advantage of chain formation is that in terms of location it avoids robots getting themselves or target lost. In addition, chain formation provides robustness against partial failures. However, this approach is not ideal to retrieve multiple items in a short time. The main reason is that most robots idle remaining static in the formation instead of searching or collecting items. On the other hand, the following approach, virtual pheromone, creates dynamic chains where all robots are busy transporting items.

2.3.2 Virtual Pheromone

Virtual pheromone is inspired on the stigmergy self-coordination performed by ants in nature (Simon et al., 1989). This is achieved by the ants leaving a pheromone trail between food and nest. This trail is reinforced by other ants that follow it. This communication mechanism allows to have the best food collection rate because it allows the colony to find the shortest path between the food and the source. In addition, virtual pheromone allows to create a computational grid (Payton et al., 2001, 2005). In robotics, virtual pheromone is used instead of chemical pheromone because chemical pheromone is expensive, experiments take time because residuals must be constantly removed and the chemical pheromone is not environment friendly.

In (Payton et al., 2001, 2005) the authors proposed an implementation of virtual pheromone for foraging task. Robots are equipped with infra-red emitters and sensors the allow them to broadcast three field messages. The first field defines pheromone type. A hops integer counter is the second field that indicates number of robots reached by the message. And the last one is data field that can carry vectors.

The robots navigate the environment by using a gas expansion model. This model consists of all robots start gathered in a single spot and then, slowly the robots start to cover the environment by maintaining a specific distance between the robots. The first robot to find the target starts to broadcast messages. Whenever a robot receives a message, it broadcasts the same message again and decreases the hop number by one. If a robot receives two or more messages at the same time, this robot broadcasts the message with higher hop counter.

Applications of the previous approach are the following (Payton et al., 2003). The first one consists in creating a static chain between items source and home area. To achieve this, robots only need to move towards the robots with highest hop counter. The second application is to use message type field to indicate robots to perform different actions like moving towards the wall to avoid any possible threads. In the third application, a victim or intruder position can be tracked by the user. The user only needs to follow the direction provided by the robots that are facing the target path direction.

Ducatelle et al. (2011a) implemented a similar virtual pheromone approach with a few changes to the contents of the message. In this case messages have two fields. The first one is an array than contains all distances that the message has travelled from robot to robot. And the second field includes age of update or sequence number that works in similar way than the hop counter in the previous approach. If a robot receives two messages at the same time it gives a higher priority to the most recent message. If the two are recent then priority is given to the message with the shorter distance.

Two types of navigation were implemented with the previous approach. The first consisted on navigation of a single robot where, in similar way as the previous approach, a single robot goes and comes back between the items source and the home area by following the members in the chain. The second is swarm navigation, in contrast of the previous one, all robots move back and forth between items source and home creating a dynamic chain as shown in Figure 2.3.

If a robot loses communication with other members two alternative actions were presented. In the first alternative the robot remains stationary at the same spot until other robots come closer. The second alternative consists of the robot navigating the environment with random movements. Results show that second variant performs better than first because stationary robot have the risk of never finding the target again.

A different approach of virtual pheromone is the implementation of a heterogeneous swarm where static aerial robots guide terrestrial robots (Ducatelle et al., 2010, 2011b). Aerial robots are positioned all over the environment making a aerial grid. Aerial robots send messages to

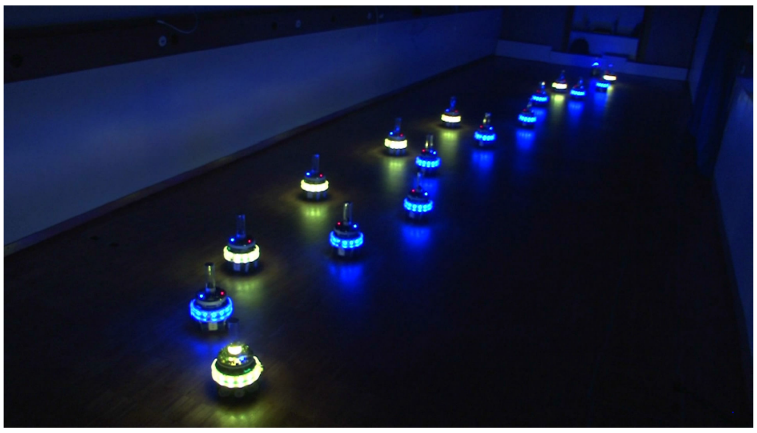


Figure 2.3: Dynamic chain formation Ducatelle et al. (2011a). Robots transporting items from the nest to the home area in a chain formation shape.

terrestrial robot with directions to travel to. Terrestrial robots provide feedback to the aerial feedback telling them if there are any obstacles. Aerial robots adjust directions and send them again. As explained before with virtual pheromone it is possible to find the shortest path and keep robots idle transporting items. However, communication is key to achieve this emergent behaviour. Without communication robots become obstacles that do not contribute to the chain formation.

2.3.3 Task Partitioning

According to (Ratnieks and Anderson, 1999), task partitioning is defined as the division of a discreet task into different individuals. In contrast to division of labour which consists of individuals allocation to different tasks.

Task partitioning in nature is composed of foragers searching for food and collectors that receive food from the foragers to transport it to the nest. Food can be transported in three different ways: direct, indirect or a combination of both. Foragers ants perform indirect transfer where ants cut leafs and let them fall on the ground (Hart and Ratnieks, 2001). Honey bees foragers provide the nectar directly to the collector in the direct transfer (Johnson, 2010). Ponerine ants foragers focus on stinging preys and they can either leave it on the ground or transfer it directly to other ants (Schmickl and Karsai, 2014). The main advantage of task partitioning is that the swarm self-regulates task allocation according to the collection performance. For instance if there is a high quantity of food foragers number will increase. On the other hand, if there are not enough collectors, foragers will switch task and become collectors.

Task partitioning for foraging swarm robotics was initially implemented to avoid bottlenecks near the home area (Goldberg and Matarie, 2001; Pini et al., 2011; Brutschy et al., 2014). Later, task partitioning was used to reduce dead-reckoning error.

In (Goldberg and Matarie, 2001) the authors proposed to divide the main task into two different fixed subtasks where item transference takes place in the nest. In the first one, a group of robots transports items to somewhere near the nest. In the second one, a robot is previously assigned to wait near to the nest in order to receive the items from the foragers. The main disadvantage is that task allocation for robots is done by the user and even though interference number decrease, performance decreases due to transference waiting times.

In order to reduce waiting times and predefined task allocations seen in the previous implementation, authors in (Pini et al., 2011; Brutschy et al., 2014) implemented a controller that allows robots to switch task and find the correct robots distribution. An exchange zone is previously positioned between home area and items source as shown in Figure 2.4. A robot that takes an item from the source transports it to the exchange zone and waits until a second robot approaches and receives the item. The second robot then takes the item to the home area. After leaving the item in the home area, the second robot comes back to the exchange zone and waits for another robot to bring an item from the items source. If a robot waits for too long in the exchange zone, then the robot switches state and takes the item by itself to the home area. This is done in order to prevent deadlock where robots wait forever in the exchange zone. Results show that interference number decreased and performance increased.

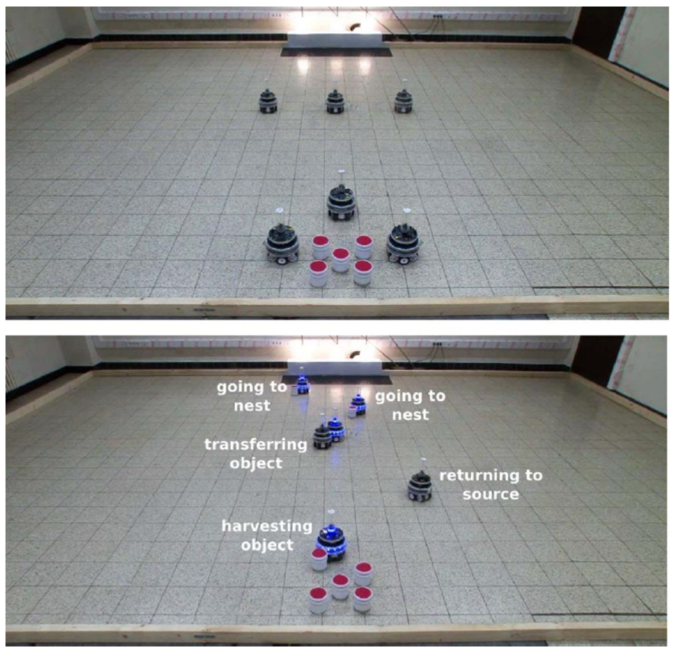


Figure 2.4: Task partitioning with foot-bot robots through transference interface by Pini et al. (2014) .

Robots use odometry for navigation because it offers the advantages of being low cost and it can be used indoors or places where there is lack of access to a global positioning service. However, odometry is susceptible to dead-reckoning error. Errors can be systematic (introduced by the robot design itself) and/or nonsystematic (introduced by the interaction between the robot and the environment) (Carlson and Murphy, 2005). Even though, systematic error can be decreased like Borenstein and Feng (1996) did, nonsystematic errors are hard to reduce due to their non-linearity nature. Desert ants, *Cataglyphis*, experience something similar as odometry error. If the length of the ant legs is modified, ants experience a higher probability of getting lost at the moment of retrieving food to the nest (Wittlinger and Wolf, 2006).

In (Pini et al., 2014) the authors used task partitioning to decrease dead-reckoning error. For their simulations and experiments the authors used a robot platform which has a bias towards the left direction. The main problem with dead-reckoning error is that it accumulates with the distance travelled. Therefore, the longer distance the robot travels the bigger the error will be. Navigation and landmark tracking are inaccurate because of this error. The error is reduced by dividing the total travelling distance between home area and items source into multiple smaller segments. However. the main limitation of this implementation is that number of subtasks is previously given by user which only would work for a specific robots number, error model and environment given.

In biology, a similar phenomenon can be found in *Laisus* ants where navigation accuracy increases by dividing the task into smaller subtasks (Ratnieks and Anderson, 1999). Each ant knows well its own subtask and the probability of getting lost and dying decreases.

In (Pini et al., 2013) the previous limitations are handled by implementing a cost function mapping to simulated robots. In this approach robots divide the total travelling distance into a fixed set of segments and a random cost is randomly assigned to each segment. Costs for each segment will change according to a function that evaluates the performance of the robot by itself at the moment of using that segment as travelling distance. Eventually robots converge to the distance segment that provides the best performance. The downside of this approach are that it presents slow convergence and discretization is fixed.

As explained before tasks partitioning has the advantage of low interference between robots and error reduction. Emergent behaviours are achieved without any explicit communication.

Evolutionary task partitioning

Inspired on leafcutting ants in nature (Hart and Ratnieks, 2001), the authors in (Ferrante et al., 2015) evolved controllers for simulated robots to achieve task partitioning similar to the one seen in leafcutting ants.

In nature three different behaviours emerge for leafcutting ants (Hart and Ratnieks, 2001). The first one is called *generalist* in which ants take leafs from source directly to nest. In second one ants take leafs from source and drop them forming a leaf cache, this behaviour is referred as *dropper*. And finally, *collector* ants collect ants from cache and takes them to the nest. These three behaviours are shown in Figure 2.5.

In Ferrante et al. (2015) the authors firstly analysed task partitioning for two different scenarios (see figure 2.5) with a group of four robots using pre-evolved dropper, collector or generalist strategies. The first scenario is a flat arena in which the group of robots performs best when there are only generalist robots. And for the second scenario, it had a slope, that slows transition of ants over it. Results shown that when there are two collectors and two droppers the highest performance is achieved. This shows that task partitioning performs better in an environment in which there are slopes that slows robots movement and where dropping items helps to increase performance.

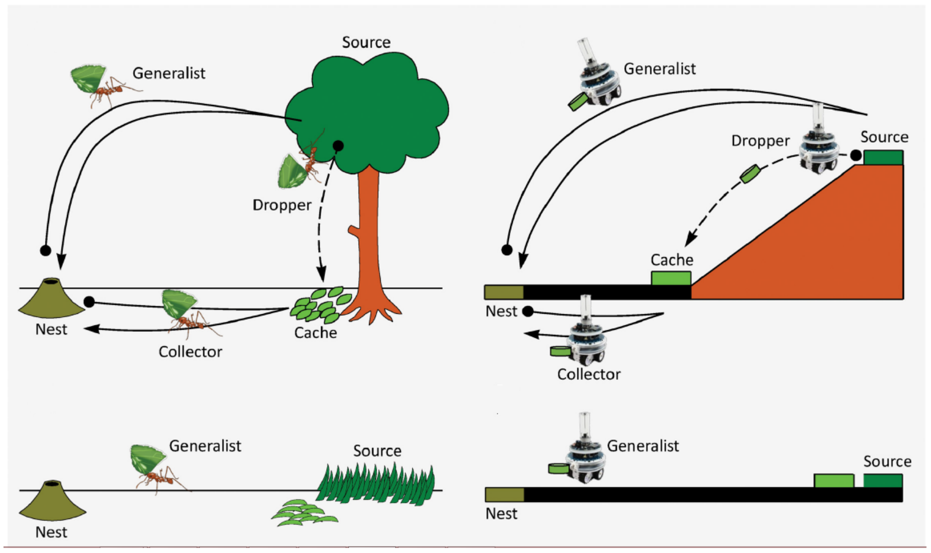


Figure 2.5: Task partitioning in nature and robotics Ferrante et al. (2015) . Top figure illustrates partitioning strategies and bottom figure show non-partitioning strategies with leafcutting ants (left side) and robots (right side).

The main advantages of this approach are that there is no need to determine proportion of individuals (collectors and droppers) and there is no task allocation mechanisms. However, controllers might not translate well in physical robots due to the reality gap and the controller evolution with physical robots can take a long time.

2.4 Conclusion

Due to their high degree of robot redundancy, swarm robotics systems were thought to be inherently robust. However, (Winfield et al., 2005; Winfield and Nembrini, 2006) showed that this is not always the case and for some scenarios, specific partial failures in robots can lead to a task degradation or, in worst case, the task not be achieved.

This chapter gave an overview of chain formation implementations for foraging found in literature (chain formation, virtual pheromone and task partitioning). Even though it has been shown that chain formation and virtual pheromone are resilient to faults, communication is key to achieve the formations. In task partitioning there is communication and formations emerge. In addition, error produced by dead-reckoning error decreases.

This thesis presents Dynamic Partitioning Strategy (DPS) a penalty and reward mechanism where robots divide the foraging task into multiple components autonomously and increases the item collection accuracy. This is achieved without communication between robots.

Chapter 3

Experimental Methodology

3.1 Introduction

In this chapter, the methodology followed for the experimental work in this thesis is presented. First, tools used to run the experiments and analyse the results are described. Task partitioning strategies are explained next. Then, parameters for the experiments, and possible outputs from experiments are explained. Finally, the statistical tools used to analyse the experimental results are discussed.

3.2 Tools

ARGoS (Pinciroli et al., 2011) is a simulator that allows the users to run experiments for large-scale number of robots. Controllers are written in C++ for different robot platforms such as the e-puck, the foot-bot and the psi-swarm. The robot platforms used for the experiments in this thesis are the foot-bot and the psi-swarm robots both of them shown in Figure 3.1 and described below.

For the experiments in Chapter 4, a simulated version of the foot-bot robot platform is used (Dorigo et al., 2011) (Figure 3.1a). This robot has the following sensors and actuator: a camera, proximity sensors, light sensors, ground sensors and a gripper.

For the experiments in Chapter 5 and 6, a simulated and real version of the psi-swarm robot platform are used (Hilder et al., 2014) (Figure 3.1b). This robot has the following sensors: infra-red sensors, a ground colour sensor to detect the colour of the ground surface. Blue-tooth communication is used to exchange messages with the server.

Since the psi-swarm robot platform does not have a camera, a virtual camera is used instead to aid the robot to detect the items. The virtual camera consists of a tracking system (Figure 3.2) that retrieves the position of an ArUco tag (Garrido-Jurado et al., 2014) attached on top of the robot. A blue-tooth signal is sent to the robot to let it know it has found an item. This is

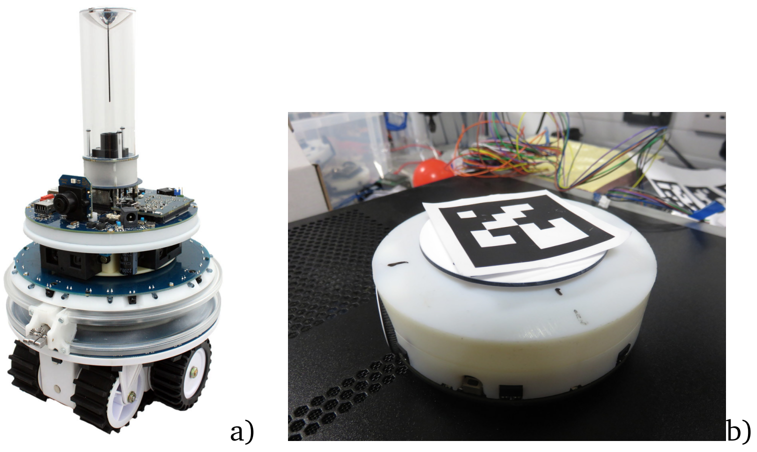


Figure 3.1: The robot platforms used for the experiments are the foot-bot (a) and the psi-swarm (b).

the only function of the tracking server.

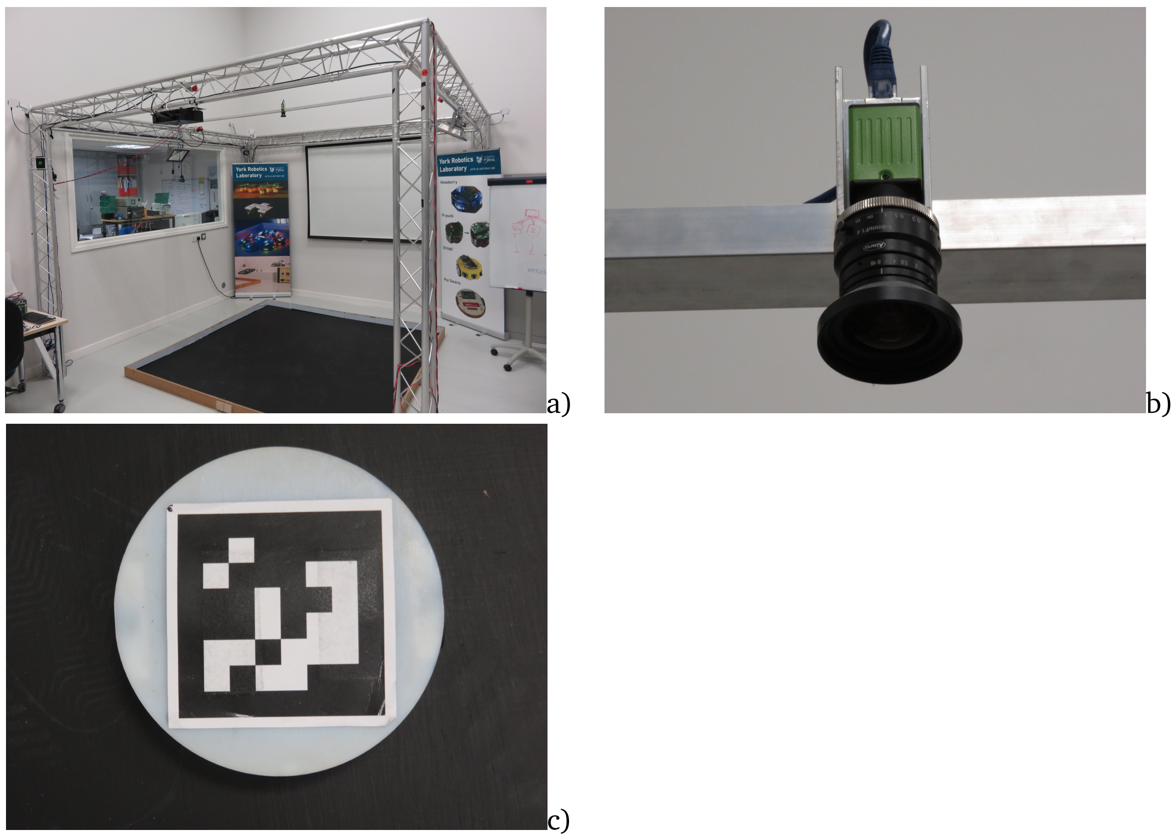


Figure 3.2: ArUco tag tracking system and their components: supporting structure (a), camera (b) and tag (c).

For statistical analysis, Spartan is used (Alden et al., 2014), a tool that has different statistical analysis techniques including consistency, parameter robustness and latin-hypercube analysis. These techniques help to provide a better understanding about the experiments shown for the approach proposed in this thesis. The techniques will be described in more detail later in this section.

All source code for the controller used in the experiments detailed in the paper can be found on line at `https://github.com/edgarbuchanan/dps` for experiments in Chapter 4 and `https://github.com/edgarbuchanan/psiswarm_task_partitioning` for experiments in Chapter 5.

3.3 Foraging task

The task used as a case study in this thesis is foraging. A group of robots explore the environment searching for items to collect. After an item is found, the position is recorded by the robot and then, the robot transports the items towards the home area. Robots use dead reckoning as navigation but due to systematic and unsystematic errors, real time navigation is inaccurate leading to error in the landmark positioning estimation. Error accumulates as the robot travels which affects the estimated item position. In consequence, when the robot attempts to go back to where it found the last item, it reaches a different location.

A virtual beacon guides the robots towards the home area. The arena, shown in Figure 3.3, has a rectangular shape defined by *w x l* and the distance between the home area and the items source (*d*). In this work, all the items are located in a single fixed spot and the items are considered to be unlimited. The home area is shown as a grey rectangular shape just in front of the beacon opposite where the items are located.

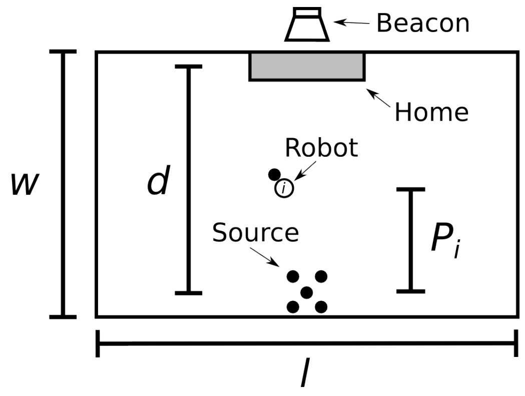


Figure 3.3: The arena has a rectangular shape defined by *w x l*. The distance between home area and items source is represented by *d*. P_i represents the individual partition length for robot i.

3.4 Partitioning strategies

In this thesis four task partitioning foraging strategies are examined.

1) Non-partitioning Strategy (NPS): In NPS, the robots take the items directly from the item source to the home area as shown in Figure 3.4a. There are no subtasks in this strategy, therefore there is no transference of items between robots. As explained in the incoming chapters, in this method the robots experience the highest amount of error because the robots are transporting the item the total travelling distance (P_T).

2) Static Partitioning Strategy (SPS): All the robots transport the item the same distance P_S in this strategy as shown in Figure 3.4b (Pini et al., 2014). Then, the item is exchanged from robot to robot until the item reaches the home area. The distance P_S is predefined by the user and constant across the experiment length.

3) Cost Partitioning Strategy (CPS): In this strategy, the P_T is discretised into a set of smaller lengths (Pini et al., 2013). The robots measure the amount of time that it takes to transports an item from the point of collection until a new item is collected for each partition in the set. The partition P_i that provides the least amount of time of collection has higher probability of being chosen next.

4) Dynamic Partitioning Strategy (DPS): DPS is the strategy proposed in this thesis where each robot changes its own individual partition length, using a penalty and reward mechanism. Every time a robot finds an item P_i increases. If the robot does not find an item then P_i is decreased. With CPS and DPS each robot has its own P_i according to the performance of the robot itself (Figure 3.4c).

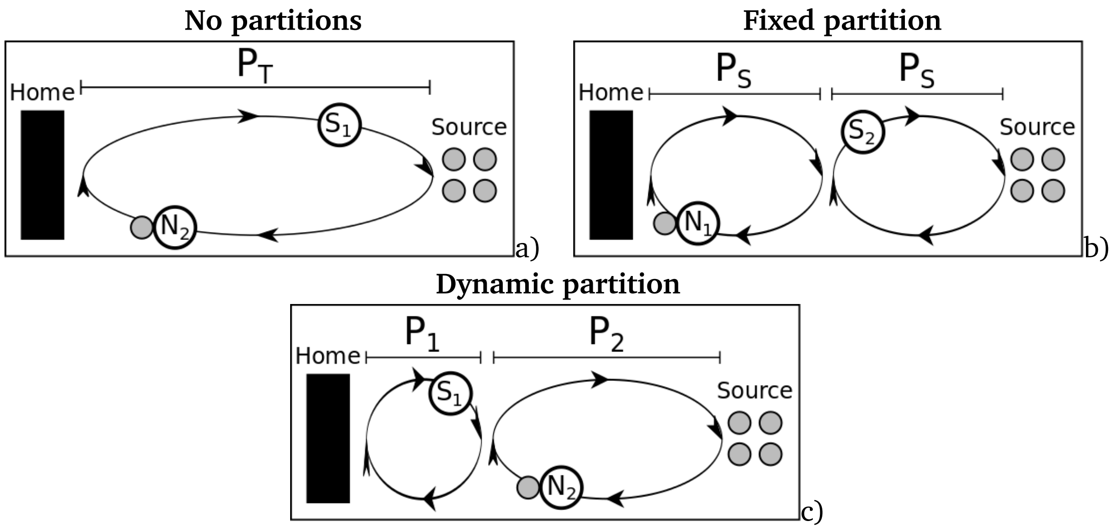


Figure 3.4: Partitioning methods. Non-partitioning method (a) where robots take the items directly to the nest. Static partitioning method (b) where the partition length is predefined and fixed. Dynamic partitioning method (c) the partition length changes for each robot according to its performance.

More details about each strategy can be found in the next chapters such as the Finite State Machine (FSM) for each strategy, P_i estimation, advantages and disadvantages of each strategy.

3.5 Item transference

In biology, food transference in task partitioning with insects is achieved by direct, indirect or a combination of both. The main advantage of direct transfer, food losses from thieves or other types of losses are reduced (Ratnieks and Anderson, 1999). For instance, 53% percent of the leaves are lost due to the indirect transfer with *Atta sexdens* ants. However, this has low impact on the leaf collection because this happens early in the sequence and little time has invested. In addition, leaves supply is considered unlimited.

In this thesis robots exchange items with direct (Chapter 4 and 6) and indirect transfer (Chapter 5 and 6).

In indirect transference, also known as bucket brigade item transference, the robots exchange the items directly with each other as shown in Figure 3.5a. First, a robot takes an item from the source and travels towards the home area for a limited distance (P_3) and then it stops to wait for a second robot to hand the item over to (W_3). Then, a second robot approaches to the robot waiting and receives the item. Lastly, the second robot transports the item towards the nest (N_1).

In the *indirect transference*, the robots leave the items on the ground instead of waiting like in the previous method, as shown in Figure 3.5b. Firstly, after a robot takes an item from the items source, it travels towards the home area (N_2) for a limited distance (P_3) and drops the item. The items remain in that position until a different robot picks it up. Then, the robot (S_1) returns to where it found the last item.

In terms of real world applications direct transfer could be used for transporting delicate materials (e.g. chemicals) where the highest priority is to guarantee the item reaches the home area. Indirect transfer can be used for not delicate material (e.g rubbish) and immediate transportation to home area is not priority.

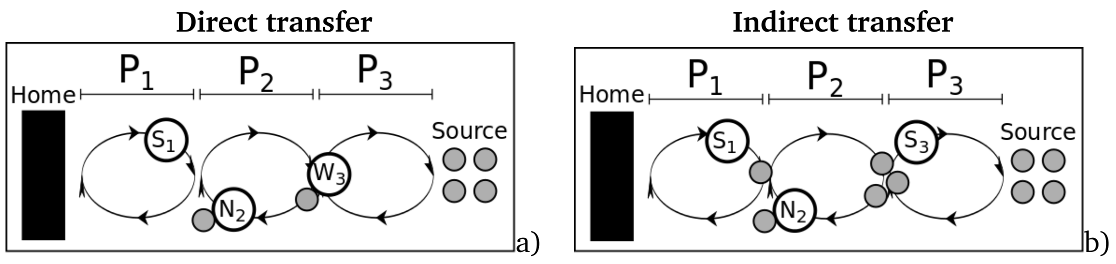


Figure 3.5: Transference mechanisms: direct transfer (a) and indirect transfer (b). In the direct transfer, robot $W3$ waits for robot $N2$ to retrieve the item. In the indirect transfer, robot $S3$ leaves the items and returns to collect more items.

3.6 Macro and micro experimental perspective

The experiments shown in this thesis are classified into two perspective: macro (global) and micro (local) perspectives.

Experiments from a macro perspective (Figure 3.6a) provide an understanding of the swarm as a whole. An example metric for this perspective is the *total items collected* where only the total amount of items is considered and individual robot contribution is ignored. Chapters 4 and 5 show results from a global perspective.

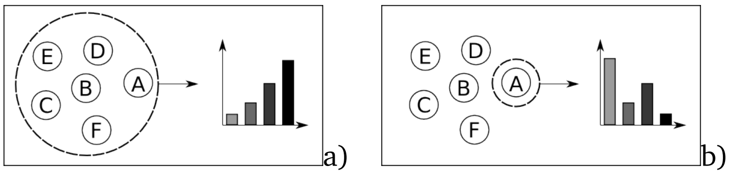


Figure 3.6: Macro and micro perspective studies. The macro perspective represents the results from the swarm as a whole (a). The micro perspective shows results for each robot within the swarm (b).

Experiments from a micro perspective (Figure 3.6b) provide the behaviours of each robot within the swarm. The P_i is a parameter that represent the partition length for robot i. A micro perspective study is important when models differ from each robot. Chapter 6 show results from this perspective.

3.7 Parameters and outputs

3.7.1 Parameters

The parameters that are used for the experiments presented in this thesis are *initial partition length* (P_i^0), *simulation length*, *arena size*, *swarm size*, *d*, *transaction time*, *dead reckoning noise*, α, $epsilon$ and *memory factor* described below.

a) Initial partition length (P_i^0): Initial distance that a robot travels from where it found an item for the first time, towards the nest measured in meters. In addition, all robots start with the same P_i^0.

b) Simulation length: Duration of the experiment until it stops.

c) Arena size: The arena has a rectangular shape of *w* by *l*, and these dimensions will change for each experiment. It is important to mention that the arena layout does not change therefore the items source is always located at the bottom of the arena and the nest in the top.

d) Swarm size: Number of robots performing the task. The minimum *swarm size* is 2 because at least a pair of robots are required to have task partitioning.

e) d: Distance between the home area and the items source.

f) Transaction time: Amount of time that it takes for a pair of robots to the hand an item over to each other.

g) Dead-reckoning noise: The *dead-reckoning noise* is modelled as a small drift towards the left or right direction according to each robot platform. In order to create this drift the right and left speed motor are increased or decreased by a certain amount. The value that defines the amount of change of speed to each motor is collected from different distributions. A different noise model is used for each robot platform and they are described in more detail in Chapters 4 and 5.

h) α: The parameter α is used to regulate proportionally the amount of change of the penalty and the reward as shown in Equation 3.1. The penalty is given by $k\alpha$ and the reward by $k(1-\alpha)$. α regulates the weight for the penalty and the reward.

$$P_i(t) = \begin{cases} P_i(t-1) + k(1-\alpha) & \text{if item found} \\ P_i(t-1) - k\alpha & \text{if item not found} \end{cases} \tag{3.1}$$

i) ϵ: Parameter that regulates the ratio between exploration and exploitation for CPS.

j) Memory factor: This parameter defines the speed of learning.

3.7.2 Outputs

The outputs from the experiments in this thesis are *final* $\tilde{P}$, *items collected, individual performance, subtasks number, convergence speed, collection ratio* and *explore ratio* described below.

a) Final $\tilde{P}$: Last median of $\tilde{P}$ when the experiment stops.

b) Individual performance: Total number of items collected at the end of the simulation divided by the swarm size. An item is considered collected whenever it reaches the nest.

c) Subtasks number: Number of times a single object has been picked up by a robot. Therefore this defines how many subtasks compose the main task.

d) Total items collected: Number of items collected at the end of the experiment.

e) Convergence speed: In order to measure the time that the swarm takes to reach the steady state the equation $|\tilde{P}(t) - \tilde{P}(t-3600)| < 0.05$ m is used where $\tilde{P}(t)$ represents the median of P_i at time t for all the runs. This equation evaluates that within an hour (3600 seconds) the differences between medians in that time window is smaller than 0.05 m which means that for our experiments the swarm reached a steady state. Finally, in order to measure stability an average of the coefficient of variation (CV), which is a measure of dispersion defined as the ratio of the standard deviation to the mean, is calculated from the samples once the swarm reaches the steady state.

Table 3.1: Statistical tests and techniques used in each section in this thesis

Test	Section (s)
Mann-Whitney U-test	Section 4.3.2
Vargha-Delaney A-test	Sections 4.3.3, 5.4 and 7.2
Consistency analysis	Section 3.8
Robustness analysis	Sections 4.3.3 and 7.2
Partial Rank Correlation Coefficients	Sections 4.3.3 and 5.4
Extended Fourier Amplitude Sampling Test	Section 5.4
Approximate Bayesian Computation	Section 5.4
Non-dominated Sorting Genetic Algorithm II	Section 5.4

f) Collection ratio: The *collection ratio* is measured with $\frac{I_F}{(I_F+I_L)}$ where I_F is a counter that records the number of times an item is found since the beginning of the simulation and I_L is a counter that that records the number of times an item is not found.

g) Explore ratio: The *explore ratio* represents the amount of time spent by the swarm in the *explore* state or not transporting items. The *explore ratio* is measured with $\frac{T_E}{T_T}$ where T_E represents the total time for all the robots spent in the *explore* state and T_T is the *simulation length* multiplied by the *swarm size*.

h) Social entropy: *Social entropy* is a metric the measures diversity in robot and it was initially introduced by Balch (1998) inspired from information theory. We use this metric to measure the homogeneity across the swarm. Social entropy $H(R)$ is calculated with Equation 3.2 where M is the number of subsets (faulty and non-faulty), p_i is the proportion of agents in each subset i and R represent the group of robots.

$$H(R) = -\sum_{i=1}^{M} p_i log_2(p_i) \tag{3.2}$$

3.8 Statistical tests and techniques

In this section the statistical tests used to analyse the results from the experiments in this thesis are described. Table 3.1 shows where each test and technique is used in this thesis. For more information about each technique, please refer (Alden et al., 2014) and (Alden et al., 2018).

The Mann-Whitney U-test is used to test the null hypothesis (O) that the results from one strategy and a second strategy are samples from continuous distributions with equal medians. In case that the null hypothesis is rejected then the one-tailed directional alternative hypothesis (A) is tested using the left tail. A hypothesis is rejected when the p score is greater than 0.05.

The Vargha-Delaney A test is a non-parametric effect magnitude test that can be used to indicate the difference between two distributions (Delaney and Vargha, 2000). The more different the distributions are, the closer the score is to 0 and 1. The data does not need to transformed as suggested in Neumann et al. (2015) because the amount of time that it takes for our variables to change is greater than the length of the tick we are using for our simulations (0.1s). The consistency and robustness analyses described below utilise the Vargha Delaney A-Test.

1) Consistency analysis: The consistency analysis technique allows to identify the sample size that minimizes the effect of aleatory uncertainty caused by inherent stochastic variation within non-deterministic simulations. In this technique 20 distributions are compared with the Vargha-Delaney test for different number of runs. The value of 130 for the experiments in Chapter 4 and 180 for the experiments in Chapter 5 and 6 as sample size have an A-Test score below 0.56 which suggests that the aleatory uncertainty in the outputs has been mitigated and also avoids over-fitting the experiments with a larger sample size (Figure 3.7).

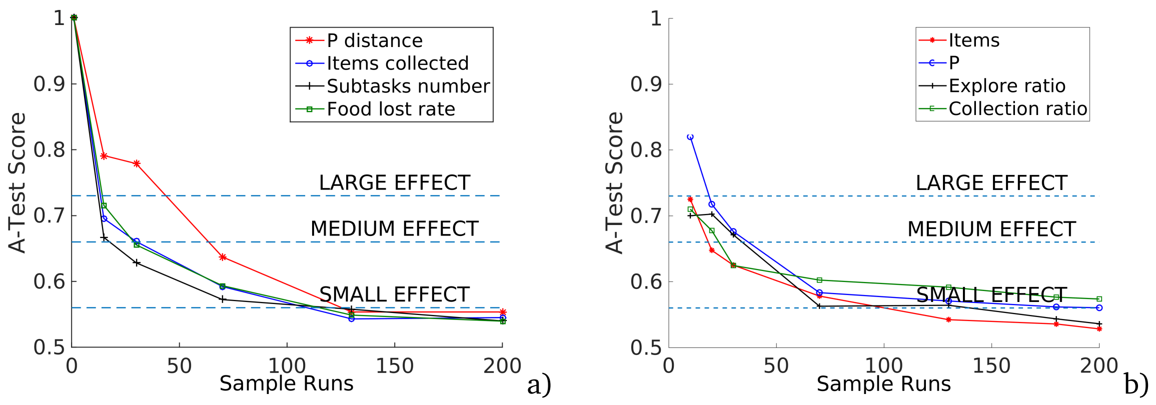


Figure 3.7: Consistency analyses that show that the sample size values of 130 (a) and 180 (b) are large enough to avoid aleatory uncertainty effect (A-Test lower than 0.56) in the outputs without over-fitting the results from the simulations for the foot-bot (a) and psi-swarm (b).

2) Robustness analysis: The robustness analysis technique helps to identify how robust the simulation behaviours are to an alteration of a single parameter value, with respect to the baseline values. In order to have a large effect the A-test comparison between the perturbed and baseline result distributions needs to be less than 0.27 or greater than 0.73. A parameter has no contribution to the system if this parameter is between the previous interval.

3) Partial Rank Correlation Coefficients: The Partial Rank Correlation Coefficients (PRCC) are used here in order to identify the degrees of dependency between each parameter. This is done by performing a Latin-hypercube sampling across parameters space and the number of samples are 1000. The main difference between random sampling and Latin-hypercube sampling is that with the second it is possible to increase reliability that the entire space is covered adequately. Relationship between parameter and outputs is measured with Partial Rank Correlation Coefficients. The correlation coefficients provides a measure of influence of a single parameter with a single output. Strong correlations (close to 1 or -1) correspond to influential parameters over its respective output in spite of non-linearity introduced by other parameters.

4) Extended Fourier Amplitude Sampling Test: The extended Fourier Amplitude Sampling Test (eFast) technique partitions the output variance caused by a parameter perturbation between the input parameters. This provides a strong indication of the influence of each parameter over each output. In addition, a ’dummy’ parameter is introduced, which does not have impact on the emulation and helps to measure if the contribution by a single parameter is statistically significant compared with the dummy parameter used as baseline. *Si* represents the fraction of variance accounted by that parameter and *STi* represents the fraction of variance accounted by that parameter and any higher-order or non-linear effects between this parameter and others. A parameter has a strong influence over a specific output if the *Si* and *STi* values are high.

5) Approximate Bayesian Computation: The Approximate Bayesian Computation (easyABC) provides a parameter distribution layout that gives rise up a specific expected behaviour. In other other words, easyABC provides a density measure where the values for a specific parameter lay that exhibit a specific output. If the graph is skewed towards a specific side this means that values closer to that side have higher probability to provide the expected output. If the layout is somewhat like a normal distribution, the value of that parameter is not important for the expected output.

6) Non-dominated Sorting Genetic Algorithm II: Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a technique that explores the entire parameters space in order to maximize and/or minimize multiple outputs. Once, all the solutions population converge (or meet the specific convergence criteria) this set of solutions is referred as pareto front.

Chapter 4

Dynamic partitioning strategy

4.1 Introduction

In this chapter, the different properties of the Dynamic Partitioning strategy are described. In the simulations in this chapter the robots divide the tasks by using direct transfer, where robots exchange items directly with each other. A screenshot of one of the experiments presented in this chapter is shown in Figure 4.1.

Experiments shown in this chapter include study of the convergence to solution DPS, performance comparison between DPS and NPS, SPS and CPS and a study of the penalty and reward ratio. In addition, in Section 4.4 the penalty and reward mechanism is implemented to an aggregation task to demonstrate that this mechanism is not limited to only foraging tasks.

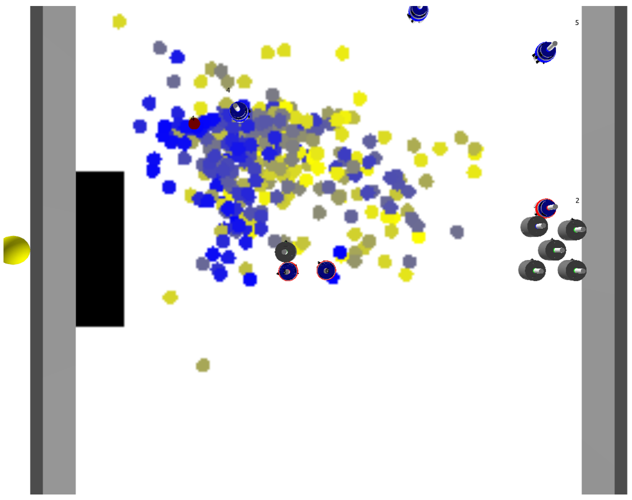


Figure 4.1: Screnshot of one of the experiments presented in this chapter. The robot platform used is the foot-bot robot. Items are represented as grey cylinders. Home area is the black region located on the left hand side of the enviroment and the items source is located on the right hand of the arena. The circles on the ground represent where the item was exchanged where the light colour (yellow) represents the oldest exchange and the darkest colour (blue) represents the most recent exchange.

4.2 Task Partitioning

In this section, the finite state machines used as controllers, capabilities and limitations are described for each strategy. Firstly NPS and the dead-reckoning noise model are described. Then, SPS and CPS are presented with their respective limitations. Finally, DPS is defined and a comparison with the other strategies is presented.

4.2.1 Non-partitioning strategy

In NPS the robots take the items from the source directly to the home area. This means that there are no partitions and the robots attempt to return directly to the source. The FSM used for this strategy is composed of four states: *explore*, *go to nest*, *go to source* and *neighbourhood exploration*. Each state is shown in the Figure 4.2 and described below.

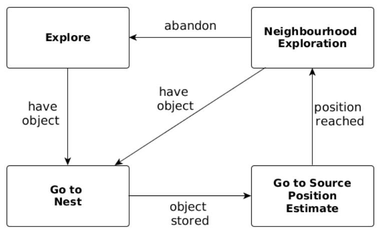


Figure 4.2: Nonpartitioning strategy (NPS) finite state machine, after Pini et al. (2014), composed by four states: *explore, go to nest, go to source* and *neighbourhood exploration*.

First, the robots begin in the *explore* state in which all robots are searching the environment for items to collect. The exploration consists of robots making linear movements with a 5% probability of making a random turn for each simulation tick. The exploration consists of a motor schema based navigation (Arkin, 1987) with two behaviours: stay on path and avoid obstacle. Once an item is within the range of vision of the robot's camera (0.5 m), the robot approaches to the item, records the position and enters to the *go to nest* state.

In the *go to nest* state, the robot travels towards the home area by following the beacon. Through the use of light sensors, the robot aligns itself towards the strongest source of light and then moves in a straight line towards the beacon. The robot senses that it has reached the home area when its ground sensors detect a change in colour on the surface from grey to white. The robot then enters the *go to source* state.

The *go to source* state consists of the robot going back to where it found the last item, assuming that it will find more items in that position. The robot uses dead-reckoning to guide itself to reach the location, where it will enter the *neighbourhood exploration* state.

The *neighbourhood exploration* state enables the robot to find the item again in the case that the item changed position, or the target position drifted slightly from the original position. This state is similar to the *explore* state, however in this case, the robots explore a circle of radius 0.5 m centred on the target position. This exploration takes up to three minutes, and if the robot fails to find any item within this time window, then it will transition to the *explore* state. However, if an item is found then the robot will return the *go to nest* state.

In an ideal scenario the items will be continuously retrieved by the robots from the source. However, in reality, robots will be subject to different common factors such as gear wear, wheel/track slippage and objects stuck in the gears (Carlson and Murphy, 2005), which will produce an error due to the dead-reckoning noise that accumulates as the robot moves. Therefore, the longer the distance the robot travels the greater the error will be, thus increasing the probability of losing the items source target due to the drift between the estimated position and the actual position.

In order to decrease the probability of the robots of losing the positions of the items, Pini et al. (2014) propose to divide the total travelling distance into smaller segments. This method is described in the following section.

4.2.2 Static Partitioning Strategy

SPS consists of dividing the total travelling distance between the items source and the home area into smaller segments. Robots exchange items in a chain until the items reach the nest. The FSM for this strategy is similar to that of NPS; the main difference is that instead of travelling all the way to the nest after finding an item the robots stop at a specific distance Partition Length (P) and wait for another robot to hand the item over, as in Figure 4.3. The value of P for the experiments shown by Pini et al. (2014) was 2.0 m.

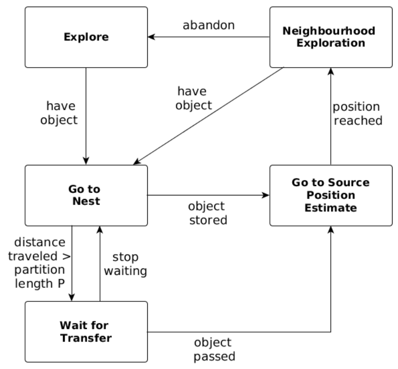


Figure 4.3: Static partitioning strategy (SPS) finite state machine adds the *wait for transfer* state to the NPS FSM.

In the *waiting for transfer* state, a robot waits for another robot to approach and take the item from them. Robots wait for up to three minutes, and if no robot approaches then the robot with the item transitions to the *go to nest* state and takes the item directly to the nest. This is done to prevent deadlocks where all robots are simultaneously in the *waiting for transfer* state and waiting for each other; but it carries a penalty in that the robot is highly likely to become lost as it returns to the source. However, if another robot comes and takes the object then the first robot goes back to where it found the last item in the *go to source* state. This sequence repeats until the item reaches the nest. When two robots meet to exchange the item there is a delay called the transaction time. This transaction time simulates the time that it takes to the robots to grip and release the object and its value is 15 seconds for the experiments shown in this chapter.

It should be noted that both robots, the robot that gives and the one that takes the item, share the same FSM. The robot that receives the item, which can be either in the *explore* or *neighbourhood exploration* states, switches to the *go to nest* state.

The main limitation of this approach is that P is predefined by the user for that specific model of physical robot. Therefore, if the dead-reckoning noise in the robots or the robot platform changes, the output of the item collection might be different from expected. As a result, this can lead to a performance degradation.

4.2.3 Cost-based Partitioning Strategy

CPS is an autonomous task partitioning strategy based on cost estimation. In this strategy the robots divide the total travelling distance between the items source and the home area into multiple smaller lengths. In addition, each robot assigns a cost to each length. Every time a robot grips an object it measures the time, cost, that it takes to grip a new item. The length with the lowest cost has the higher probability to be chosen next in the next iteration.

For cost estimation, each robot i divides the total travelling distance between the items source and the home area d into N multiple smaller lengths. N is given by the Equation 4.1 where Δ represents the discretization step defining the granularity of the landscape. This process is illustrated in Figure 4.4a.

$$N = [\frac{d}{\Delta}] \tag{4.1}$$

In contrast to SPS and DPS, with CPS each robot has a set of predefined lengths referred as $\mathbb{L}$. The distance that comprise each member L_j of $\mathbb{L}$ is given by Equation 4.2.

$$L_j = \Delta \cdot j \text{ with } j \in \{1, 2, ..., N-1\} \tag{4.2}$$

Initially, a random initial cost $\hat{C}_j$ is assigned to each robot i. Every time an object is gripped by the robot, the robot starts to measure the time that it takes to find another item again. After another object is found, $\hat{C}_j$ is updated according to Equation 4.3.

$$\hat{C}_j = (1-\alpha)\hat{C}'_j + \alpha\bar{C} \tag{4.3}$$

where $\alpha \in (0, 1]$ is the memory factor, $\hat{C}'_j$ is the previous estimated cost and $\bar{C}$ is the actual cost from given by

$$\bar{C} = \frac{d}{L_i}(\Theta_n + \Theta_g) + \Theta_{rw} \tag{4.4}$$

where Θ_n is the cost related with navigation, Θ_g is the gripping cost and Θ_{rw} is the time spent exploring the environment.

An example for a set of cost estimations is shown in Figure 4.4b where for this case L_3 has the lowest cost, therefore this partition length has the greater probability of being selected by the $\epsilon - greedy$ algorithm.

An $\epsilon - greedy$ algorithm is used to select the next L_j as P_i for each robot i. The ϵ parameter defines the balance between exploration and exploitation. As ϵ gets closer to 0 the exploitation increases which means that the global best solution is identified quicker but robots are less resilient to changes. On the other hand, as ϵ gets closer to 1, it takes a longer time for the robot to identify the global best solution because the robot is exploring all the possible solutions but the robot is more resilient to changes in the environment or changes to the noise levels.

An example for a set of cost estimations is shown in Figure 4.4b where for this case L_3 has the lowest cost, therefore this partition length has the greater probability of being selected by the $\epsilon - greedy$ algorithm.

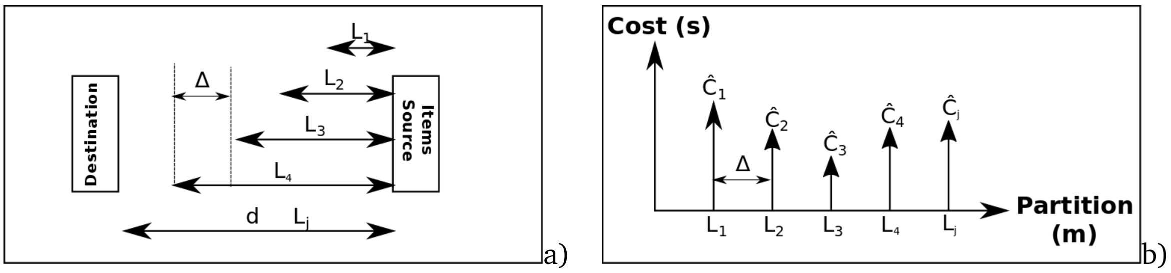


Figure 4.4: Cost-based Partitioning Strategy (CPS) diagrams. Diagram in (a) illustrates how d is decomposed into a set of multiple partition lengths. Example of how costs ($\hat{C}_j$) can change for each partition length L_j are shown in the diagram (b).

4.2.4 Dynamic Partitioning Strategy

DPS, proposed in this thesis, allows the robots to adapt to dead-reckoning noise by establishing a suitable value of P automatically. The FSM for this strategy is shown in Figure 4.5.

Each robot is rewarded/penalized every time it finds/loses an item during the *neighbourhood exploration* state. The robot i increases its value of P_i by a specific predefined fixed amount (G), if the item is found. This is shown in Equation 4.5, where $P_i(t)$ is the value of P_i after the

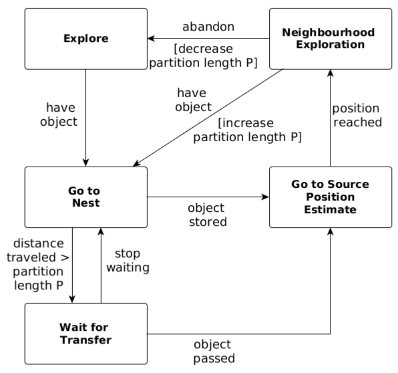

Explore
Neighbourhood Exploration
abandon
[decrease partition length P]
have object
have object
[increase partition length P]
position reached
Go to Nest
Go to Source Position Estimate
object stored
distance traveled > partition length P
stop waiting
Wait for Transfer
object passed

Figure 4.5: Dynamic partitioning strategy (DPS) finite state machine where P changes after the *neighbourhood exploration* state.

application of the penalty or reward, and $P_i(t-1)$ is the value beforehand.

$$P_i(t) = \begin{cases} P_i(t-1) + G & \text{if item found} \\ P_i(t-1) - G & \text{if item not found} \end{cases} \tag{4.5}$$

The value of $P_i(t)$ converges to a distance where the robots experience a specific probability of finding objects. This rate is defined by the ratio of penalty to reward, which is fixed in Equation 4.5 but is explained in more detail in the Section 4.3.3. The speed of convergence is controlled by the value of G and is studied in Section 4.3.1.

In comparison with the previous approach, CPS, the differences that DPS present are the following. Firstly, the initial P_i is defined by the user, although it can be randomly assigned, in contrast with CPS always randomly assigned. Secondly, in DPS, P_i changes according to whether or not the robot i finds an item when it goes back to the source as defined in Equation 4.2. In CPS, P_i is selected according to a $\epsilon - greedy$ algorithm giving priority to lengths with lower costs. Thirdly, the final P_i, for DPS, is defined mainly by the ratio between the penalty and rewards, as shown in Section 4.3.3. In CPS, the final P_i selected is the one that has the lowest cost. Finally, for CPS, in order to find the best P_i it is necessary to explore the entire landscape. On the other hand, with DPS it is only necessary to explore one side of the landscape as shown in Section 4.3.1.

An important characteristic that both algorithms share, CPS and DPS, is that if the environment or the inherent error in the robots changes, P_i for each robot adapts to these changes regardless of the values selected for the parameters. In contrast to SPS where is not as flexible where performance of the swarm changes with these changes.

For work in this chapter, direct transfer is used, as described above, in preference to indirect transfer, where the robots deposit the items instead of waiting to transfer them. This is due to the fact that work in this chapter is based on the previous work of Pini et al. (2014), use was made of direct transfer.

Table 4.1: Arena size and simulation parameters for convergence experiments

Parameter	Value
Arena width *w*	4.5 m
Arena length *l*	6.7 m
Source-to-nest distance (d)	4.0 m
Number of robots	6
α	0.5
P_i^0	0.5 m, 3.5 m
G	0.05 m, 0.1 m, 0.2 m

4.3 Experiments

In this section, experiments demonstrate the different capabilities of the strategy proposed in thesis using the foot-bot robot. First, the convergence to solution is studied with different values of G and P_i^0. Then, the total items collected and the collection ratio are analysed for the three different strategies with different errors. Finally, the reward/penalty mechanism is studied in detail by analysing the effect on different values for each parameter and their effect on the performance in the swarm. Additional supplementary material can be found online at `http://www.york.ac.uk/robot-lab/dtp/`.

4.3.1 Convergence to Solution

In this section, experiments demonstrate how the swarm is able to adapt to the dead-reckoning noise in the robots to allow a continuous item collection with a fixed collection ratio.

For the experiments in this section, all robots share the same dead reckoning noise, therefore all the robots of the swarm should adapt in a similar way. The arena size is 4.5 x 6.17 m with *d* of 2.0 m. In addition, all the robots start with the same P_i^0 of 0.5 m or 3.5 m for each experiment. This is done to show convergence to similar solutions from both sides for $\tilde{P}$. P_i converges to the final solution by changing with a fixed value G as shown in Equation 3.1. In order to illustrate the effect of G on the convergence different values are tried. Parameters for the simulation and dimensions for the Normal environment are shown in Table 4.1.

As shown in the standard deviations within the runs over time in Figure 4.6, $\tilde{P}$ converges to a distance regardless of P_i^0. P_i converges to a steady state where the rate at which each robot gets lost is related to the reward/penalty ratio. This ratio will be described more in detail in Section 4.3.3. The steady state in these experiments, with the given arena size and noise model and with a penalty/reward ratio of 1:1, is a distance close to 2.4 m. Although, as for P_i^0 = 0.5 final $\tilde{P}$ is slightly shorter, because it takes longer to converge, as shown in Table 4.1, and it has not reached that distance. Since the robots are travelling smaller P_i more robots are required to create the chain between home area and items source, therefore more time is spent by robots waiting than collecting items (this is explored more in detail in Section 4.3.3).

Convergence times and the final average CV are shown in Table 4.2.

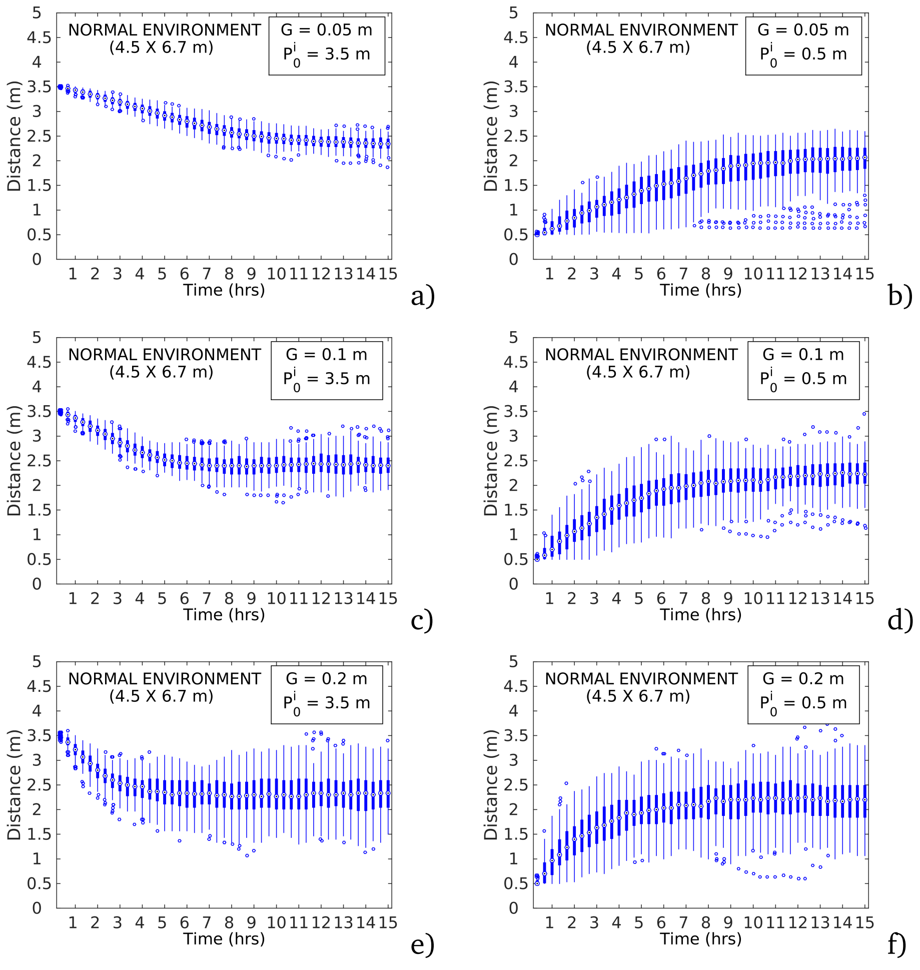


Figure 4.6: Convergence graphs to solution for two different values of P_i^0 and two different values of G. Left column graphs present robots starting with a P_i^0 of 3.5 m (a ,c ,d) and on the right column robots start with a P of 0.5 m (b, d, f). Each row presents a different value of G used for DPS .

The amount of change G to P_i is related to the speed of convergence, as shown in Figure 4.6. The convergence speed increases in relation with G however, the stability is affected and this can be seen in the value of CV.

On the other hand, as G decreases the speed of convergence will be slower but more stable because of its shorter value of CV. However, if G is too small it is harder for the robots to converge to the solution due to the fact that there are not enough robots to create the chain between home area and item source. This causes robots to get trapped with low values of P_i (Figure 4.6b).

In conclusion, DPS is able to converge to a single solution with different initial values of P_i^0 regardless the amount of error the robots are subjected to. For the rest of this section the G considered will be 0.1 m and the reason is that it provides a good convergence speed with good stability.

Table 4.2: Convergence times and final average coefficient of variation for different initial P_i^0 and G.

	P_i^0 = 3.5		P_i^0 = 0.5	
G (m)	Time (hrs)	CV	Time (hrs)	CV
0.05	9.2	0.05	10.0	0.18
0.1	6.3	0.09	7.2	0.14
0.2	5.5	0.17	5.0	0.19

Table 4.3: Arena sizes and simulation parameters

Parameter	Normal	Large
Arena width W	4.5 m	6.7 m
Arena length L	6.7 m	9.9 m
Source-to-nest distance D	4.0 m	6.2 m
Experiments duration	15 hr	20 hr
Error intervals (σ)	.5σ, σ, 1.5σ	
Swarm size	2, 4, 6, 8, 10, 15, 20, 30	

4.3.2 Individual Performance Comparison Between Strategies

In this section, the four strategies will be compared with respect to the individual performance. Swarm size and error will change for each experiment.

For the experiments shown in this section there are two different arenas with different dimensions. In addition, there are three different dead-reckoning noise levels (0.5σ, σ, 1.5σ) that all the robots will be subjected to, and finally different swarm sizes are investigated. However, P_i^0 remains constant for all the experiments at 2.0 m. The reason for this is that all robots should start with the same solution as SPS. In this way, results can then be fairly compared and any variation detected in the item collection rate, when the noise is σ. Parameters for the experiments are shown in Table 4.3.

In these experiments the strategies SPS and CPS are also compared with DPS. The fixed P distance is 2.0 m for SPS. As for CPS the parameters the provided best results in (Pini et al., 2013) are chosen and they are 0 for ϵ and 0.2 as memory factor.

Results from experiments are illustrated in Figure 4.7 and results from the Mann-Whitney U-test evaluation to hypotheses are shown in Table 4.4. In Figures 4.7b, 4.7d, 4.7e and 4.7f Mann-Whitney U-test shows that there is no statistical significant improvement compared with NPS and CPS for small swarm sizes when the error is 1.5σ. The reason for this is that more robots are required in order to successfully create a chain formation between the nest and the source. For the normal environment at least 4 robots are required when the error is 1.5σ. For the large environment, 4 robots are required when the error is 0.5σ and σ, and 6 robots are required when the error is 1.5σ.

As the swarm size increases, there is a threshold when overcome the addition of more robots to the swarm affects negatively the individual performance for task partitioning strategies

(SPS, DPS and CPS) by decreasing it. This can be seen in Figures 4.7a, 4.7c and 4.7d where the individual performance decreases after the swarm size 4 for error 0.5σ and 6 for errors σ and 1.5σ, whereas the individual performance does not change for NPS. The individual performance decreases because the environment is too crowded with robots which leads them to spend more time avoiding each other than collecting items and forming the chain to collect items. Therefore, it is important to identify the appropriate swarm size that provides the best individual performance for that scenario.

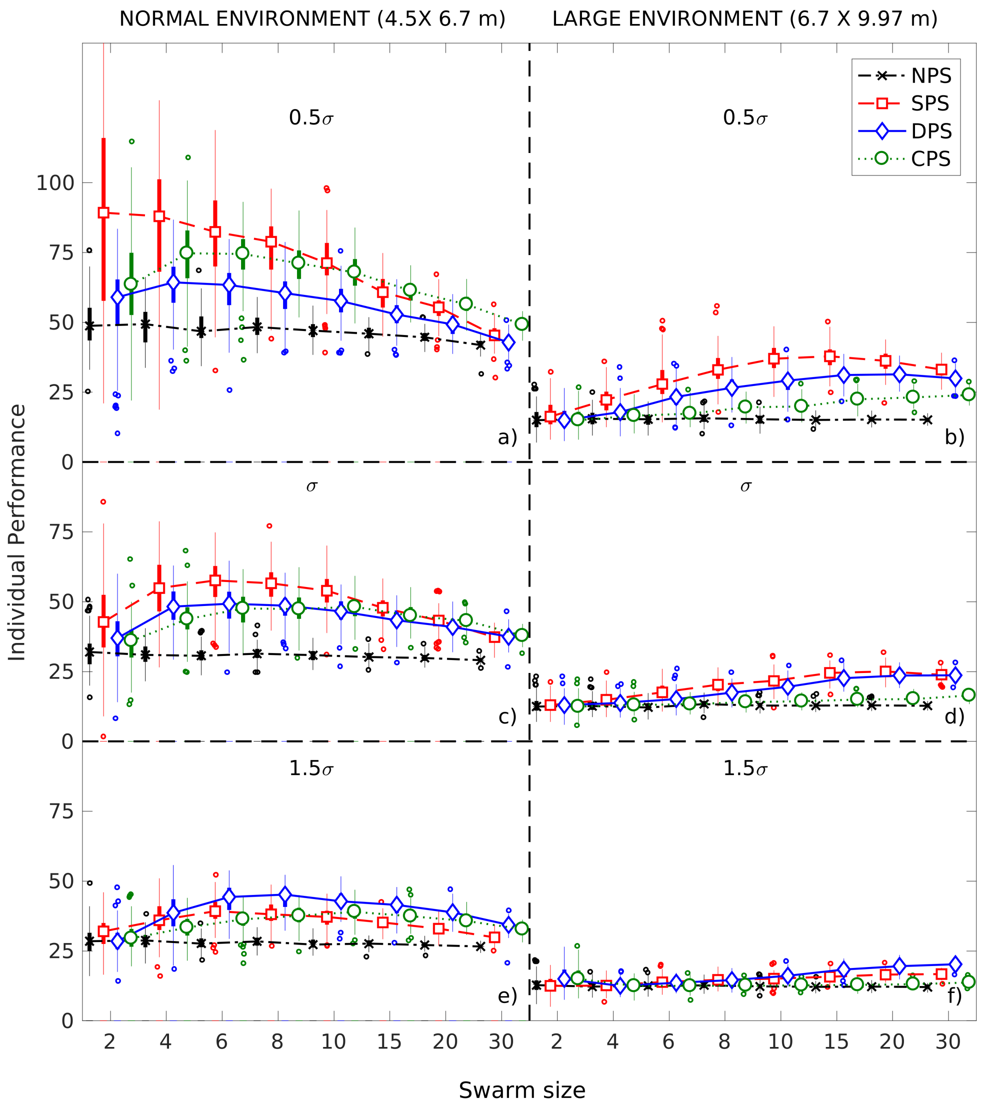


Figure 4.7: Individual performance for each strategy with different swarm size, environment size and noise. The left column represents a normal environment (4.5 X 6.7 m) and the right column represent a large environment (6.7 X 9.97 m). From top to bottom the noise levels are 0.5σ, σ and 1.5σ. For both environments, SPS performs better than the other strategies as the error decreases.

Table 4.4: Results from the Mann-Whitney U-test when the distribution for the individual performance for DPS is compared with a different strategy (NPS, SPS or CPS) with different error levels and different environments. 0 represents the null hypothesis, A represents the alternative hypothesis and x represents that the both hypotheses, null and alternative, were rejected.

			Swarm size							
Environment	Error	Strategy	2	4	6	8	10	15	20	30
Normal	0.5σ	NPS	A	A	A	A	A	A	A	A
		SPS	x	x	x	x	x	x	x	x
		CPS	x	x	x	x	x	x	x	x
	1.0σ	NPS	A	A	A	A	A	A	A	A
		SPS	x	x	x	x	x	x	x	0
		CPS	0	A	0	x	x	x	x	0
	1.5σ	NPS	0	A	A	A	A	A	A	A
		SPS	x	A	A	A	A	A	A	A
		CPS	0	A	A	A	A	A	A	A
Large	0.5	NPS	0	A	A	A	A	A	A	A
		SPS	0	x	x	x	x	x	x	0
		CPS	0	A	A	A	A	A	A	A
	1.0	NPS	0	A	A	A	A	A	A	A
		SPS	0	x	x	x	x	x	x	0
		CPS	0	A	A	A	A	A	A	A
	1.5	NPS	0	0	A	A	A	A	A	A
		SPS	0	0	0	0	A	A	A	A
		CPS	0	0	A	A	A	A	A	A

CPS has a greater individual performance than DPS when the noise level is 0.5σ, however, DPS outperforms CPS when the noise level is 1.5σ for the normal environment. DPS has a greater individual performance in the large environment for swarm sizes greater than 2 for 0.5σ and 6 robots for 1.5σ. This could allow to speculate in a similar way to (Pini et al., 2013) that that the performance of a single robot is being influenced by the performance of other robots in CPS. Furthermore, this is more evident when robots are performing direct transference. The reason is that if the robots are unable to meet again to exchange objects, this might affect the cost estimation because robots might switch to a different state. However, DPS works independently of this factor.

Item collection is lower for SPS than DPS when the error is 0.5σ and σ (Figures 4.7a, 4.7b, 4.7c and 4.7d) because the collection ratio is greater for DPS than SPS and this is described below. Although, it was expected that for the error σ both strategies should have a similar performance, in both strategies the robots start with a P_i^0 of 2 m, this is not happening due to the oscillations in the P_i described in the previous section (Figures 4.7c and 4.7d). Lastly, as the noise increases the item collection is greater for DPS than with SPS (Figures 4.7e and 4.7f).

The collection ratio is studied in more detail and shown in Figure 4.8a and 4.8b for swarm sizes of 6 and 30 respectively. The performance of SPS degrades as the noise increases for both scenarios because the robots start to lose items more frequently which means that strategy is not longer suitable for the actual conditions. This is reflected in the individual performance shown in Figure 4.7 where SPS exhibits a greater individual performance than DPS when the error is small. Conversely, performance for DPS is better than SPS because the collection ratio does not change when the error increases. Although, for small dead-reckoning noise levels SPS has a higher collection ratio than DPS and this explains the previous results where SPS has a better item collection than DPS. The collection ratio is given by ratio between the penalty/reward and this will be discussed in more detail in the next section.

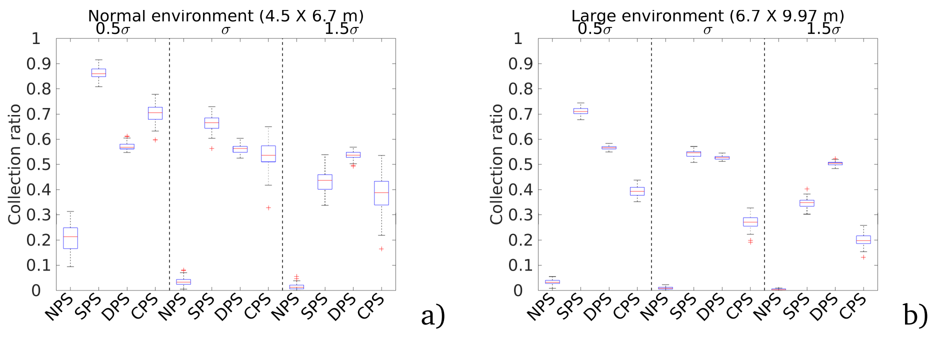


Figure 4.8: Collection ratios for NPS, SPS, DPS and CPS. Left plot reports the data for a normal environment and the right plot for a large environment. Three different errors (0.5σ, σ, 1.5σ) are reported in both plots.

The DPS's performance is greater than the other three strategies when the noise level is high

Table 4.5: Parameters used for the latin-hypercube analysis.

Input	Input range	Output
α	0-1	Final $\tilde{P}$
Dead reckoning noise (σ)	$0.5\sigma - 1.5\sigma$	Item lost rate
Distance nest-source (d)	4.0 m - 7.0 m	Subtasks number
Swarm size	2 - 30	Items collected

because it is able to maintain a constant collection ratio regardless of the error. However, for a better performance it might be necessary to find the ratio for the penalty/reward that provides the best collection. This will be described below.

4.3.3 Effect of the penalty and the reward on item collection

Even though DPS has flexibility with respect of adapting against different dead-reckoning noise levels, it still has two parameters to assign. In the previous experiments it was found that using the same value for both parameters the system provides a collection ratio close to 0.5. In this section, we explore in more detail the effect of these parameters on the system by using the α parameter.

Latin-hypercube analysis was used to provide insight into the degrees of dependency of α with other parameters. For the latin-hypercube analysis, a total of 130 samples are generated from the inputs and their respective intervals. The inputs considered for this study are α, dead-reckoning error (σ), distance between nest and source (d) and swarm size. The outputs studied are final $\tilde{P}$, collection ratio, subtasks number and total items collected. The parameters are shown in Table 4.5 and results for the α input are shown in Figure 4.9 now described.

As α increases, the probability of finding objects also increases (Figure 4.9b) because the robots are penalised for failure more than they are rewarded for success and this causes the P_i distance to get shorter as α increases (Figure 4.9a). As a result the number of times an objected is gripped by a robot increases (Figure 4.9c). Although the collection ratio increases with increasing α, this does not necessarily mean that the number of items collected increases, as shown in Figure 4.9d, where there is no correlation between these two variables. Therefore, the total items collected is not highly dependant on α which means that there are other parameters defining this output in this case. Lastly, this does not imply that the correlation between α and the item collection is not existent; this means that other parameters have a greater influence over the item collection. Correlation of α and the item collection is shown in the set of experiments.

The waiting time, time spent by a robot waiting for another robot and exchanging item, has negative effects on the rate of items collected by pushing the navigate time down after the α variable overcomes a threshold as shown in Figure 4.10. In the experiments shown in Figure 4.10, 10 robots are foraging in an arena with *w* of 4.5 m and *l* of 6.7 m and the amount

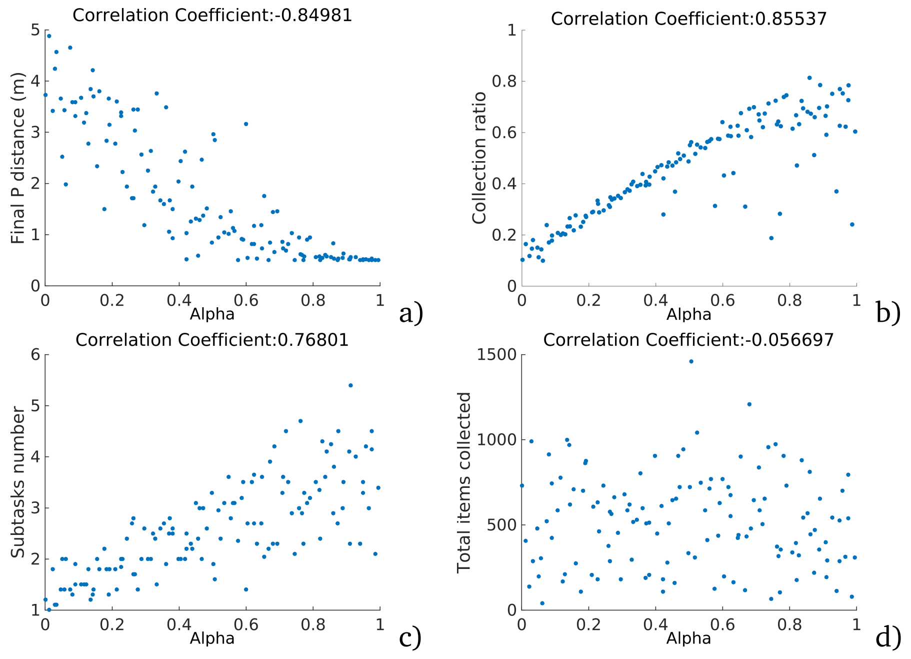


Figure 4.9: Latin-hypercube analysis of α. All outputs are strongly correlated except the total items collected which means that input is highly dependant to other parameters.

of time that robots are spending in each state is measured with different values of α. When α the robots spend more time transporting items and exploring less. However, the robots spend longer periods of time waiting because there are more subtasks as shown in Figure 4.10c. Therefore, more time is spent in transactions, which is the time that it takes for robots to hand the item over to another robots, rather than navigating. As a result it is necessary to identify what is the best value of α in order to have the best performance.

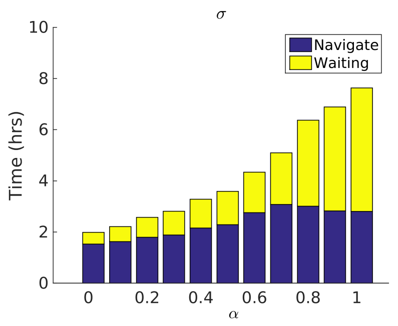
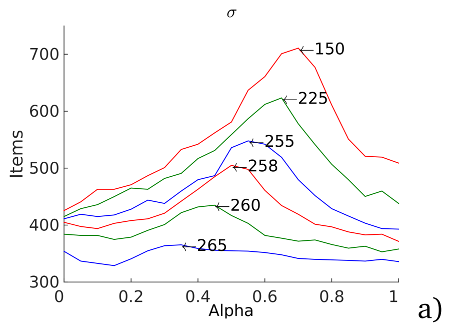


Figure 4.10: Total time robots spent for each activity. Collecting items and going back to the source (navigate), waiting for another robot or for the transaction to be completed (waiting) and exploring the environment (explore). After the value of α reaches 0.7, it starts to have negative effects on the item collection by making the robots wait longer times than navigate.

The value of α that provides the highest item collection is highly dependant on the transaction time, the swarm size and d, although there might be other variables not studied here that might have an effect on this value. Medians are collected from experiments where robots are retrieving items in an environment with w of 4.5 m and l of 6.7 m to study the effects of α on the item collection with respect of transaction time (Figure 4.11a) and swarm size (Figure 4.11b). As the transaction time increases the best value of α decreases (Figure 4.11a). Therefore, it would be better for the robots to travel a longer distance rather to spend time waiting. On the other hand, as the transaction time gets closer to 0 the ideal value of α increases which means that robots can travel shorter distances without the item collection getting too penalized by the transaction time. Lastly, the swarm size defines how much effect the α parameter has on the item collection (Figure 4.11b). If the swarm size is small there might be not enough robots to create the chain as shown in the previous section. The minimum number of robots required changes in relation with the distance between the nest and the source.



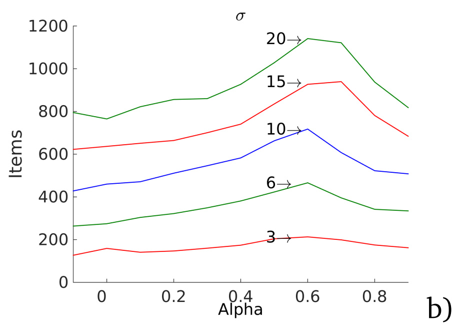


Figure 4.11: Effects of α on the item collection with respect to transaction time (Figure 4.11a) and swarm size (Figure 4.11b). As the transaction time increases the best value of α starts to drift to the left. On the other hand, the swarm size defines the impact of α in the total items collected.

The robustness analysis technique is used here to study two scenarios with d of 4 m and 6 m. Also, the swarm sizes are 1, 2 and 3 robots for the first scenario and 3, 4 and 5 robots for the second scenario. In order to have a large effect on α (A-test score less than 0.27 or greater than 0.73) in the item collection the lowest number of robots for the distance between nest and items source of 4 m is 2 robots (Figure 4.12a) and 4 robots for a distance of 6 m (Figure 4.12b). This means that for swarm sizes greater of 2 and 4 respectively selecting the appropriate value of α is critical to have the optimal performance in item collection. The baseline value chosen is 0. Finally, as the A Test score decreases, the number of items collected increases.

Besides adjusting the collection ratio, the value of α also effects the rate of item collection. In other words, as α increases the number of partitions increases, however robots spend more time during the transactions. Therefore, it is important to identify the best value of α to have the best item collection.

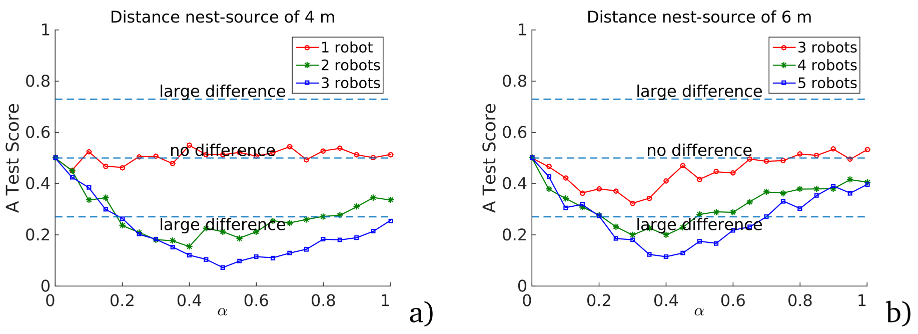


Figure 4.12: Robustness analysis for α parameter with respect of different swarm sizes and distance between nest and items source. In Figure 4.12a the distance between the nest and items source is 4 m and the lowest swarm size in order to have a large effect (A-test score less than 0.27 or greater than 0.73) in the item collection output is 2 robots. On the other hand, in Figure 4.12b the distance is 6 m and the minimum number of robots required is 4 robots.

4.4 Alpha-aggregation algorithm

In order to demonstrate that the penalty and reward mechanism proposed in this thesis is not only limited foragings tasks, in this section the penalty-reward mechanism is implemented with the $alpha$-aggregation algorithm (Nembrini, 2005).

The main objective of the aggregation algorithm is to keep a group of robots together. This is helpful when the swarm has to deal with changing environments and coherence needs to prevail between robots.

In the $alpha$-aggregation algorithm [1] the robots have the ability to communicate with each other by each robot broadcasting by its own ID. The aggregation algorithm is illustrated in Figure 4.13 and described next.

A robot starts in the *obstacle avoidance* behaviour. Robots continuously receive the IDs of their neighbours within range (radio of 0.44 m). If the number of neighbours (N) drops below the threshold *alpha* ($alpha < N$) the robot considers itself disconnected from the swarm and turns 180^o degrees and return to the *obstacle avoidance* behaviour.

Once the number of neighbours is greater or equal than $alpha$ ($N >= alpha$) that robot considers itself again connected again to the swarm. The process repeats.

In previous work the value of $alpha$ is defined by the user. In this work the value of $alpha$ changes by using the penalty and reward mechanism. This brings the benefit where each robot learns its own $alpha$.

The penalty and reward mechanism changes the value of $alpha$ according to Equation 4.6 and described next. Every time a new neighbour is found by a robot, $alpha$ increases by 0.05. This means that the robot is rewarded as more robots are added to to its neighbourhood. The

[1]The parameter α for the aggregation algorithm will be referred as $alpha$ in order to prevent confusion with the α parameter introduced in Chapter 3

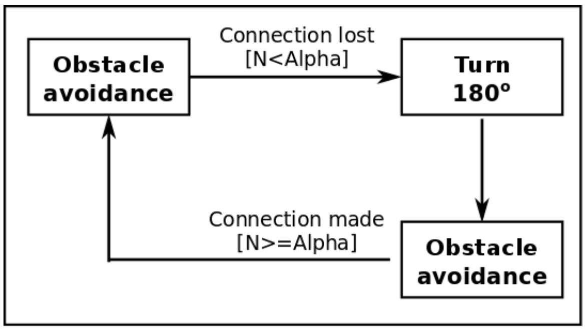


Figure 4.13: Block diagram for the *alpha*-aggregation algorithm. A robot starts in the *obstacle avoidance* behaviour. This robot switches to $turn180^o$ if the number of neighbours (N) decreases below $alpha$ and the robot considers itself disconnected from the swarm. The robot considers itself connected to the swarm when (N) is greater than $alpha$.

value of $alpha$ decreases by 0.15 if the robot loses connection from the swarm. This penalises heavily robots that isolate themselves from the swarm. Some experiments and results are the following.

$$alpha_i(t) = \begin{cases} alpha_i(t-1) + 0.05 & \text{if new neighbour found} \\ alpha_i(t-1) - 0.15 & \text{if connection lost} \end{cases} \tag{4.6}$$

The robot converge to a single $alpha$ regardless the initial value of $alpha$ in a similar way as P converged to a single P in the experiments shown in Section 4.3.1. In Figure 4.14a the robots start with an $alpha$ of 3 and the robots converge to a value close to 2. The robots in a similar way converge when starting with 1.5 (Figure 4.14c). The values are more spread because the robot struggle to get together once the swarm is across the environment. The median does not change when the robot start with a value of $alpha$ of 2 (Figure 4.14b).

The penalty and reward mechanism allows the robots to learn better solutions than with constant $alpha$ values in a similar way as shown in experiment in Section 4.3.2. In Figure 4.15 (top) the final number of neighbours for starting $alpha$ of (1.5, 2 and 3) converge to a value close to 1.4 by using the penalty and reward mechanism. A fixed value of $alpha$ provides mean number of neighbours of 1 with $alpha$ 3, 1.3 with $alpha$ 1 and 1.8 with $alpha$ of 2 as shown in Figure 4.15 (bottom).

In conclusion, the robots are able to learn the appropriate value of $alpha$ regardless of the initial $alpha$. This would become useful when robots experience different degree of error and robots required to learn different values. This explored more in detail in the next chapter.

4.5 Summary

In foraging tasks, where robots guide themselves using odometry, dead-reckoning noise is a common problem that creates a degradation in the performance of item collection. As a result,

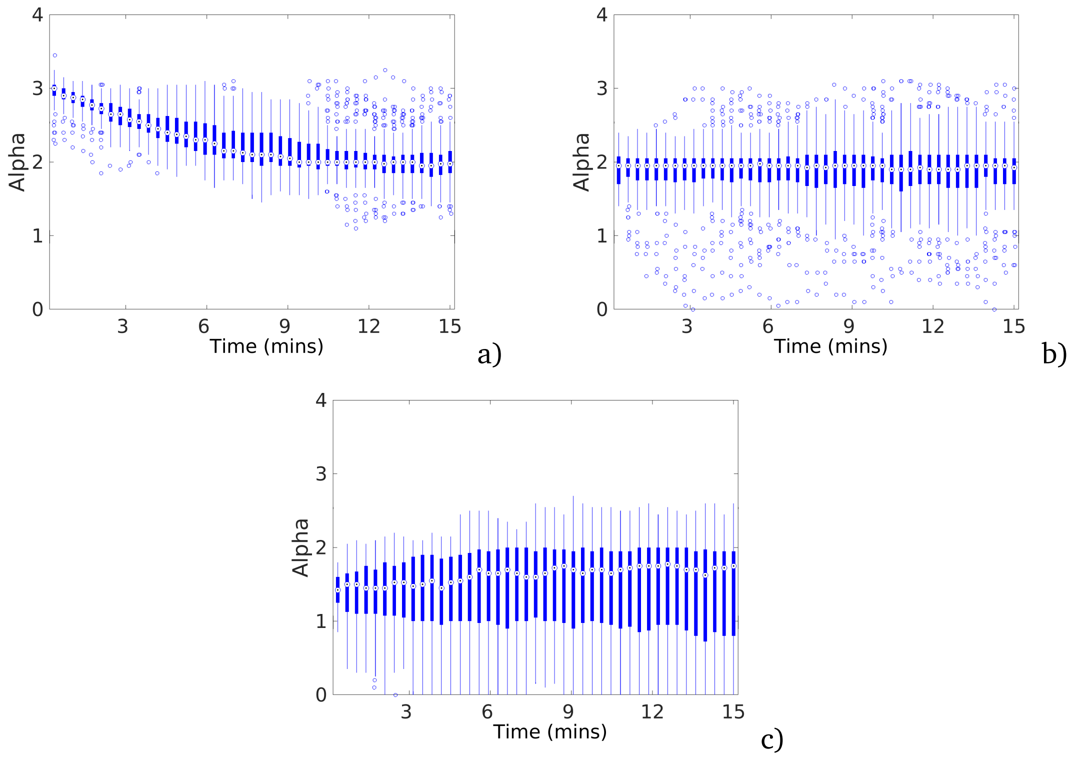


Figure 4.14: *Alpha* convergence for initial values of 3 (a), 2 (b) and 1.5 (c). The robot converge to a value close to 2 regardless of the initial value of *alpha*.

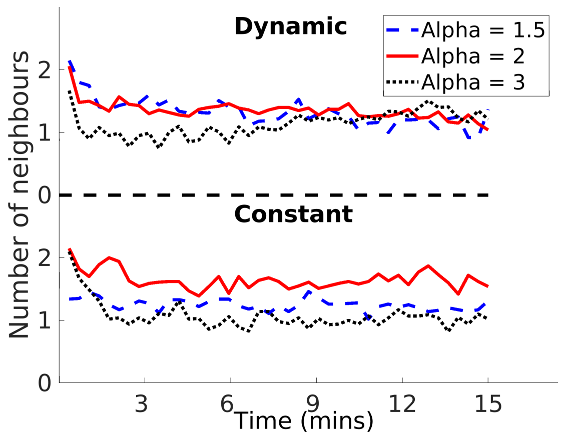


Figure 4.15: Average number of neighbours with changing *alpha* (top) and constant *alpha* (bottom). In dynamic *alpha* the robots start with different *alpha* (1.5, 2 and 3) and robot change *alpha* with the penalty and reward mechanism. The mean of number of neighbours reach 1.4 regardless of the initial *alpha*. In constrant in constant *alpha*, the robots do not change *alpha*. The number of neighbours change are 1 with *alpha* 3, 1.3 with *alpha* 1 and 1.8 with *alpha* of 2.

robots are unable to return successfully to the exact position where they found the last item. Furthermore, error accumulates as the distance between the source and the nest increases, therefore arena scalability is limited.

In order to decrease the number of robots that are unable to find the items because of this error, the total travelling distance between the nest and the source can be divided into multiple segments where the robots travel a fixed distance as proposed by Pini et al. (2014) and referred to in this thesis as Static Task Partitioning (SPS). However, it is necessary to identify the best distance that provides a good item collection performance beforehand. This distance will vary depending on the robot platform, swarm size and the dead-reckoning noise. Therefore, it would be advantageous if robots could adapt, regardless the error.

In this thesis a variation on the SPS strategy is proposed, that employs a variant on the state machine used in the SPS, a technique called Dynamic Task Partitioning (DPS). In this approach robots are penalized every time they are unable to locate items, by reducing the partition length (P) and they are rewarded when they find items by increasing the value of P.

The results in this chapter show that all the robots in the swarm are able to converge to a common solution because all the robots have the same specifications. Even though the item collection improves for bigger errors and large environments for DPS, SPS and CPS have a better performance for small errors. The reason is that the solution given for SPS and the solution found by CPS has a higher collection ratio than the solution that DPS found. Therefore, it is necessary to change the parameters in DPS in order to increase the item collection.

The ratio between the reward and the penalty controls the likelihood of the robots travelling a large enough distance that they get lost when attempting to return. If the reward is greater than the penalty then the robot will travel a longer distance then if the penalty is greater than the reward. This will result in a high and low collection ratio respectively. Nevertheless this does not mean that decreasing the distance will increase the item collection rate at all times. The item collection rate does depend also in the swarm size, transaction time and nest-source distance. There is a minimum number of robots to create the chain between the nest and the source, and this is consistent with the results from experiments in literature where it is determined that that there is a minimum number of robots in order to achieve the task such as swarm containment (Bjerknes and Winfield, 2012) and immune-inspired error detection (Lau et al., 2013).

The last section demonstrated that the penalty and reward mechanism proposed in this thesis is not only limited to foraging task. The robots were able to learn the appropriate value for the parameter that provides a constant number of neighbours.

Chapter 5

Study from a macro perspective

5.1 Introduction

In the previous chapter the different properties of Dynamic Partitioning Strategy (DPS) in terms of convergence time, item collection comparison and α were described. In Chapters 5 and 6 a more in depth study of performance of all the partitioning strategies in terms of error diversity and uniformity is described. In this chapter, The performance of each strategy is studied from a macro or global perspective, or in other words the swarm is considered as whole in this chapter. A study from a micro or local perspective is described in the next chapter.

The content of this chapter is the following. Firstly, Section 5.2 describes the error in the physical robots and the procedure to generate the simulated error model. Sensitivity analysis comparison between error diversity and university can be found in Section 5.3. Lastly, Section 5.4 gives a closing summary of the chapter.

5.2 Modelling the Error in the Psi-swarm robot platform

The psi-swarm robotic platform has an error in movement similar as the foot-bot robot platform used in the experiments in the previous chapter. The error consists in the robot having a bias of moving towards a single direction either left or right. The amount of drift and direction changes for each robot as shown in Figure 5.1.

The procedure to generate the simulated error model is the following. The error in the physical robots movement is recorded and shown as sets of trajectories (Section 5.2.1). Then, mathematical functions are generated from local curvature that describe the error in each robot (Section 5.2.2). Finally, error distribution is calibrated and replicated in simulation (Section 5.2.3). Each step is described in detail next.

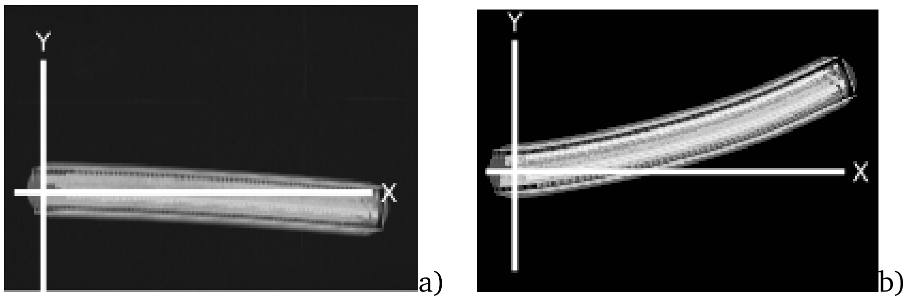


Figure 5.1: Trajectories of two different physical robots, A (a) and F (b), moving forwards for 30 seconds with a speed of 0.02 m/s.

5.2.1 Recording movement from robots

The trajectories are recorded for each robot using the ARuCo tag tracking system (Garrido-Jurado et al., 2014) where each robot has an ARuCo tag on it. Each robot is programmed to move forwards for 30 seconds with a speed of 0.02 m/s. A total of 10 repetitions are recorded for each of the 14 robots. Figure 5.2 shows the trajectories recorded for each robot. The direction and degree of error changes from robot to robot.

5.2.2 Local Curvature Function

From the points collection recorded from the trajectories, a pair of equations are generated by using curve fitting where each equation describes each coordinate over time. Equations 5.1 and 5.2 describe the X and Y coordinates respectively at any time t. Parameter P, Q, R S and T change from robot to robot.

$$X[t] = P + Qt \tag{5.1}$$

$$Y[t] = Rt^2 + St + T \tag{5.2}$$

The previous equations are used to generate a function that describes the local curvature k at any time t. The local curvature describes the amount k that a specific X and Y point deviates from a straight line at time t.

5.2.3 Simulating Error in ARGoS

The error model consists of adding noise, in a similar way as it was done in the previous chapter, to each motor as shown in Equations 5.3 and 5.4. The simulated noise μ is generated from taking a sample from a Gaussian distribution each time tick with median $k[t]$ and σ where σ changes for each robot. Every time a robot changes the speed of its motors, timer t is set to 0. This noise model recreates the bias in the robot of moving towards a single direction.

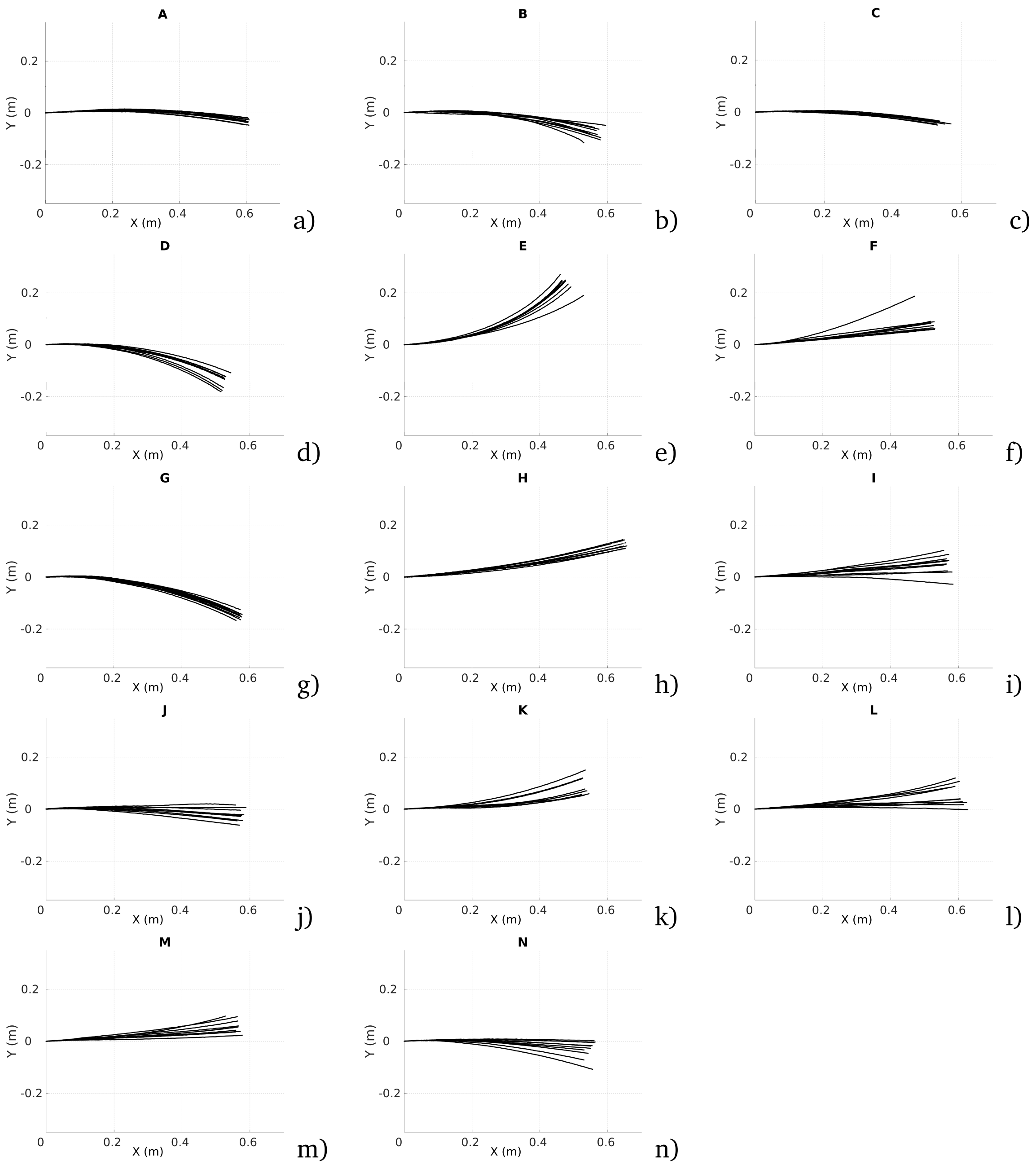


Figure 5.2: Individual trajectories of the physical psi-swarm (black lines). The letter on top of each figure represents the ID of the robot and each robot is moving forwards along the x-axis with a speed of 0.02 m/s for 30 seconds. X(m) and Y(m) represent the coordinates of the robot.

Experiments done with physical robots are replicated in the simulator and results are shown in Figure 5.3. Experiments in this thesis where each robot has different error parameters will be referred as heterogeneous error. A model of the psi-swarm used in the simulations in this chapter can be found in `https://github.com/edgarbuchanan/psiswarm_model`.

$$rightWheelSpeed = actuatedRightWheelSpeed \pm \mu * actuatedRightWheelSpeed \tag{5.3}$$

$$leftWheelSpeed = actuatedLeftWheelSpeed \pm \mu * actuatedLeftWheelSpeed \tag{5.4}$$

5.2.4 Homogeneous error

In the experiments shown in Chapters 5 and 6 two different types of errors are used, heterogeneous and homogeneous errors. The heterogeneous error was just described in the previous section. In the homogeneous error all the robots share the same error. The error model is generated from combining all the individual trajectories from the 14 robots. Homogeneous error is shown in Figure 5.4.

As far as this thesis concerns, there is no literature where diversity in terms of degrees of error is considered in experiments. Most of the research concerned about diversity across the swarm is from the evolutionary swarm robotics perspective but also reinforcement learning has been used to train specific task specialized robots (Balch, 1998; Li et al., 2003; Balch, 2005). Robot controllers can be evolved with two different methods: genetically homogeneous or genetically heterogeneous (Bongard, 2000; Mitchell A. Potter, Lisa A. Meeden, 2001; Trianni and Nolfi, 2011; Tuci, 2014; Hart et al., 2018). The homogeneous method encodes the parameters of single controller that is then cloned to each robot in the swarm. The main advantage of this method is that evolution design is simple. However, the main disadvantage is the homogeneous controllers are less robust because to some environments where role specialized robots might be required.

In the heterogeneous method each robot has its own genotype. After evolution each robot has a well-defined role in the task. The downside of this method is that the search space increases and evolution design is more complicated.

5.2.5 Indirect transfer controller

The item transfer mechanism used in Chapters 5 and 6 is indirect. The indirect item transference differs from the direct item transference in the way the item is handled between robots.

In the indirect item transference the robot drops the item on the floor after P is reached in

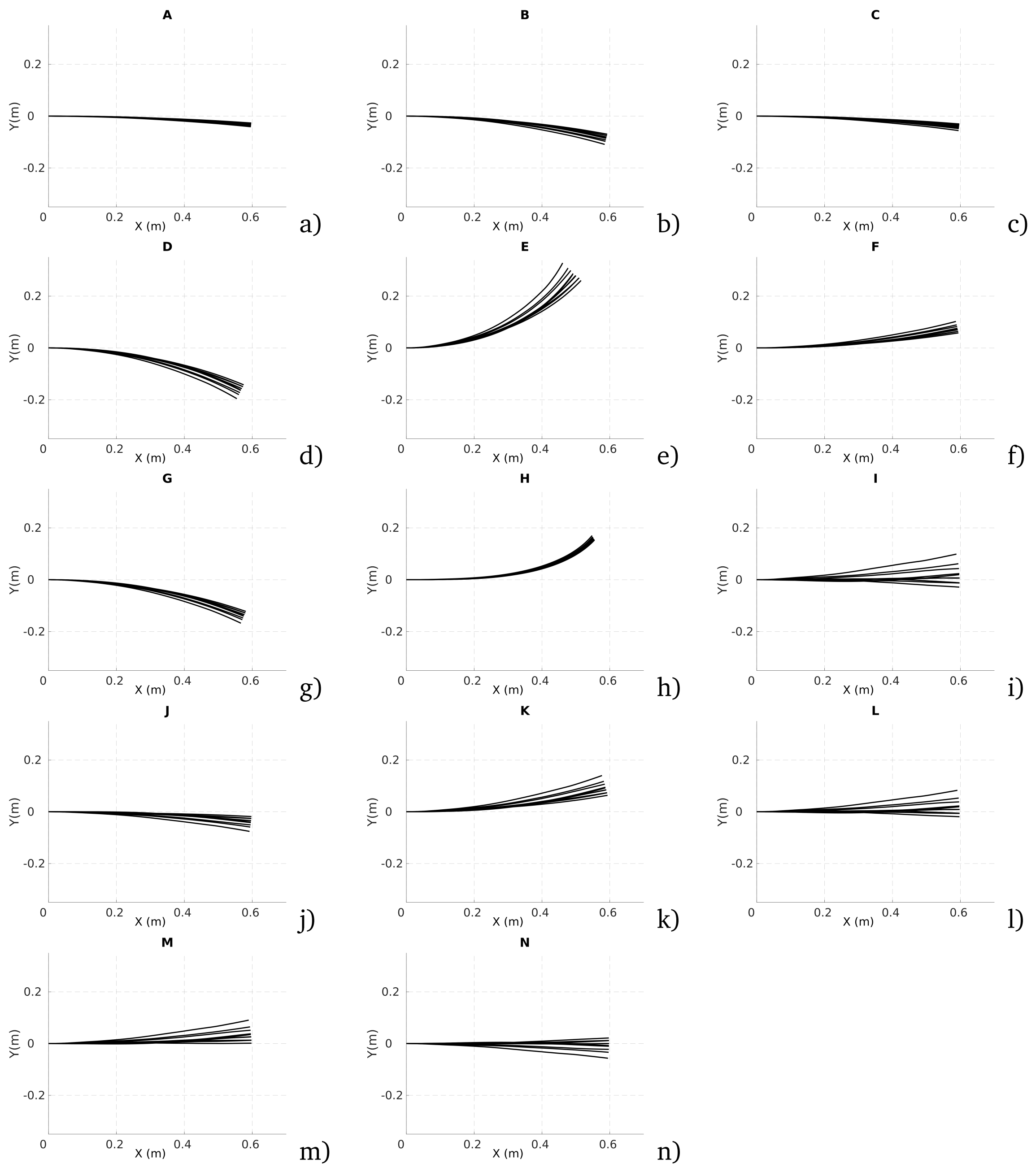


Figure 5.3: Trajectories of the simulated psi-swarms (black lines). The letter on top of each figure represents the ID of the robot and each robot moves forwards along the x-axis for 30 seconds with a speed of 0.02 m/s and 10 iterations are shown for each robot. X(m) and Y(m) represent the coordinates of the robot.

the *go to nest* state, then the robot switches to *go to source* state. In other words, the *wait for transfer* state from direct item transference is removed shown in Figure 4.5 in the previous chapter. The FSM is shown in Figure 5.5 and the controller for the psi-swarm can be found in `https://github.com/edgarbuchanan/psiswarm_task_partitioning`.

The exploration consists of a motor schema based navigation (Arkin, 1987) with two behaviours: stay on path and avoid obstacles.

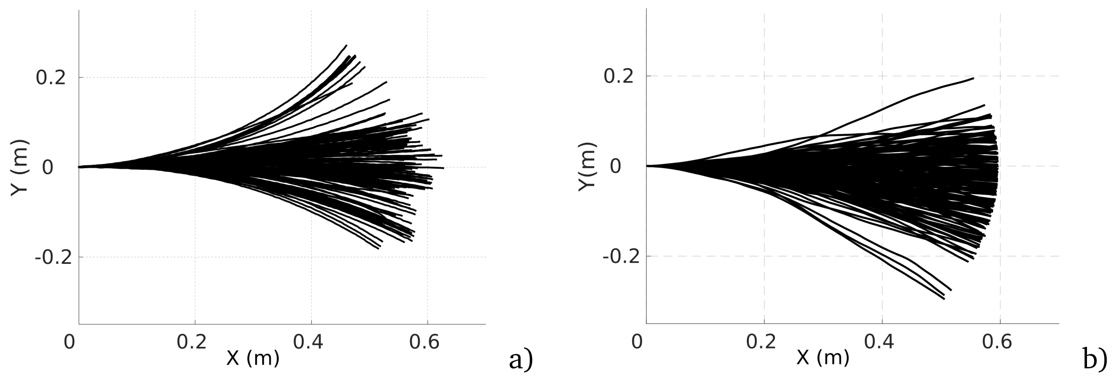
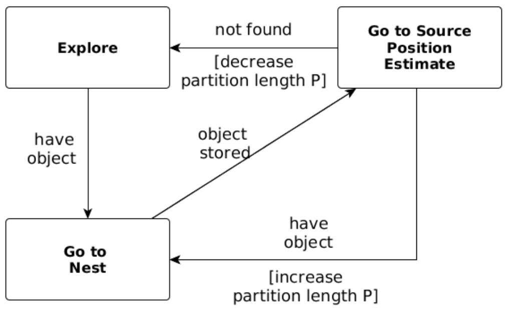


Figure 5.4: Trajectories recorded from all the robots are shown in (a) and the simulated trajectories generated for the homogeneous error are shown in (b). Each robot moves forward for 30 seconds with a speed of 0.02 m/s and 150 iterations are shown for each robot.



Explore
Go to Source Position Estimate
not found
[decrease partition length P]
have object
object stored
Go to Nest
have object
[increase partition length P]

Figure 5.5: FSM for the indirect item transfer controller. The robot drop the item after P is reached.

5.3 Experimental framework

The experimental framework followed for Chapters 5 and 6 is a top-down approach divided into three stages as shown in Figure 5.6. The first stage consists in studying the effect of implementing heterogeneous and homogeneous errors from a macro perspective (considering the swarm as a whole) by performing a sensitivity analysis to each strategy using emulation. The second stage consists in studying the effects of heterogeneous and homogeneous error from a local perspective (considering each robot as an individual) in simulations. Finally, experiments with physical robots validate results from emulation and simulations.

This chapter focuses in the first stage. The second and third stage can be found in the next chapter. As for robotic platform, the Psi-swarm robot platform is used, developed in the University of York and specifications described in Chapter 3 (Hilder et al., 2016).

The framework used for the experiments in this section is shown in Figure 5.7 and described in more detail next.

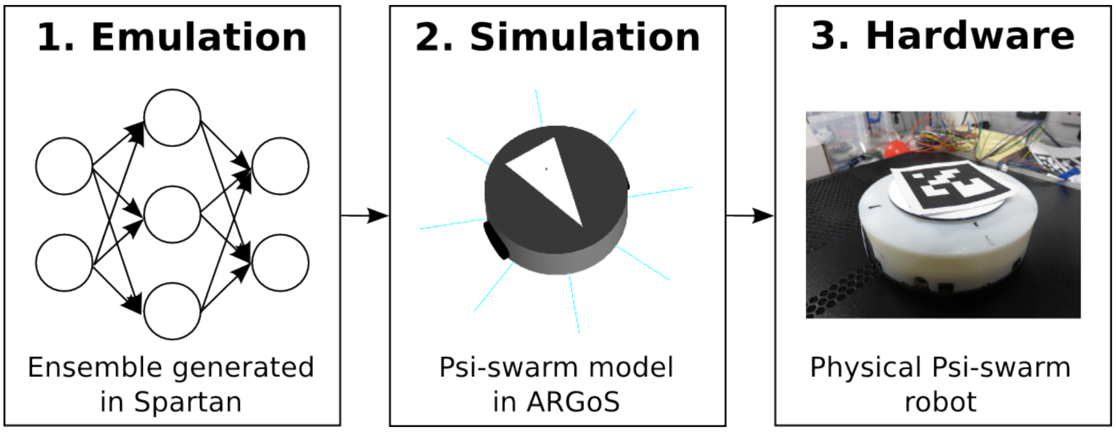


Figure 5.6: Experimental framework for Chapter 5 and 6. 1) An emulation is generated from different machine learning techniques in order to perform an enriched model analysis from a macro perspective which could not be possible to do in simulations due to constraints in time, battery span and computational power. 2) Simulations are used to study the model from a micro perspective. 3) Experiments in hardware are used to validate results shown in emulation and simulation.

5.3.1 Training data

A total of 8 datasets are generated from the outputs of the latin-hypercube sampling (Mckay et al., 1998) in simulations for each of the four strategies (NPS, SPS, CPS and DPS) and for each error type (heterogeneous and homogeneous). On average, 5 hours of simulated time takes roughly 1 minute in real time. The number of replicates needed for the experiments shown in this paper is 180 (see Section 3.8 for more information). Therefore, for a set of experiments for a single strategy it would take 3 hours in real time. The amount of time required for experiments escalates if a population of samples and/or number of generations is required.

Incorporating a combination of machine learning algorithms, an emulation is created that can be used as a surrogate for original simulation. This emulator is capable of making efficient predictions of simulation output for a given parameter set, reducing the time and resource requirements inherent in simulation due to the large number of replicates and size of the parameter space.

5.3.2 Emulator

The procedure to generate the emulator and use this to perform a predicted sensitivity analyses is the following. For each parameter in each strategy, a value range is assigned and sampled using Latin-Hypercube Sampling, that ensures adequate coverage of the parameter space (Figure 5.7i). Then, each of the eight datasets are used to train and validate each machine learning technique (Neural Network, Random Forest, General Linear Models, Support Vector Machine and Gaussian Process).

These five individual emulators are combined to form one predictive emulation, or ensemble, where predictions are generated by weighting the performance of each algorithm on a test set (Figure 5.7ii). Combining the five algorithms has been shown to increase the accuracy of prediction over using each emulator in isolation (Alden et al., 2018).

Lastly, the emulation is used to perform an enriched sensitivity analysis of the parameter space (Figure 5.7iii). To asses the degrees of dependency between parameters and outputs, Partial Rank Correlation Coefficients are calculated for each parameter-output response pair (Mckay et al., 1998). Extended Fourier Amplitude Sensitive Analysis (eFast) partitions the variance in output response between the parameters of interest, to indicate those that are highly influential in changing the emulated behaviour (Saltelli and Bollardo, 1998). Efficient Approximate Bayesian Computation (easyABC) has been used to predict the posterior distribution of each parameter in each strategy (Blum et al., 2010). Finally, to determine whether the parameters can be optimised to produced a desired behaviour, the evolutionary algorithm Non-dominated Sorting Genetic Algorithm II (NSGA-II) has been used (Deb et al., 2002). More information for each tool can be found in Section 3.8.

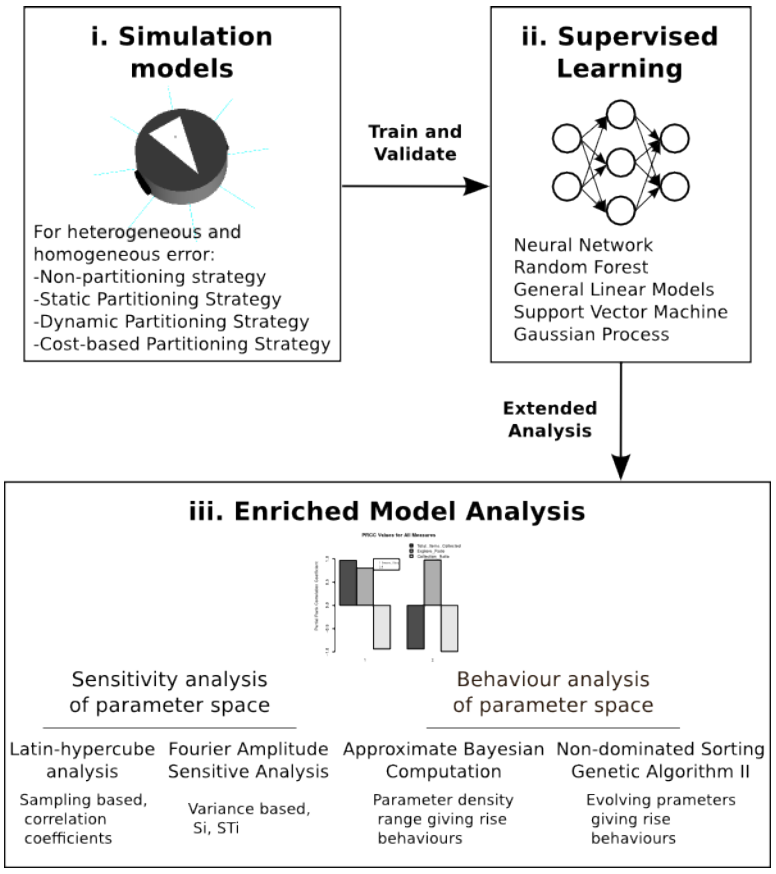

i. Simulation models
For heterogeneous and homogeneous error:
-Non-partitioning strategy
-Static Partitioning Strategy
-Dynamic Partitioning Strategy
-Cost-based Partitioning Strategy
Train and Validate
ii. Supervised Learning
Neural Network
Random Forest
General Linear Models
Support Vector Machine
Gaussian Process
Extended Analysis
iii. Enriched Model Analysis
Sensitivity analysis of parameter space
Behaviour analysis of parameter space
Latin-hypercube analysis
Sampling based, correlation coefficients
Fourier Amplitude Sensitive Analysis
Variance based, Si, STi
Approximate Bayesian Computation
Parameter density range giving rise behaviours
Non-dominated Sorting Genetic Algorithm II
Evolving prameters giving rise behaviours

Figure 5.7: From simulation to emulation block diagram. A data set is generated from latin-hypercube analysis in simulations (i). This data set is used to train and validate the ensemble composed with different machine learning techniques (ii). Different statistical tools are used to analyse the ensemble model (iii).

5.4 Sensitivity analysis

In this section a sensitivity analysis is performed to each key parameter for each strategy already discussed before (NPS, SPS, DPS and CPS) in emulation. In this way important parameters are revealed for each strategy, the understanding of the relationships between parameters and outputs with each strategy is increased and regions for parameters where maximum and minimum performance can be found.

In this chapter figures that show key results from these analyses are included, which are then discussed in more detail. However, results of the emulation training and test procedures and all statistical analyses can be found in the Appendix B.

The outputs studied for the four strategies are *explore* ratio, *collection ratio* and *total items collected*, previously introduced in Section 3.7.

The order for the strategies to be described is: NPS, SPS, DPS and CPS.

5.4.1 Nonpartitioning strategy

The first strategy to be described is the Nonpartitioning Strategy (NPS), introduced in Section 3.4, where the robots take the items directly from the items source and transport them to the home area. P_i is the same as the *distance between home area and items source (d)* and it does not change across the simulation time span.

For the NPS emulation two parameters are considered: *swarm size* and *d*. The *swarm size* is the number of robots that comprise the swarm where the range is from 2 to 14 robots. Heterogeneous and homogeneous errors are considered for the emulation. The range for *d* is 0.5 to 2.0 m. The width of the arena does not change and the home area and items source are against the walls of the arena.

Latin-hypercube analysis

The Latin-hypercube analysis (LHA) is used here in order to identify the degrees of dependency between each parameter (*swarm size* and *d*). This is done by performing a Latin-hypercube sampling across parameters space and the number of samples are 1000. The main difference between random sampling and Latin-hypercube sampling is that with the second it is possible to increase reliability that the entire space is covered adequately. Relationship between parameters and outputs is measured with Partial Rank Correlation Coefficients. The correlation coefficients provides a measure of influence of a single parameter with a single output. Strong correlations (close to 1 or -1) correspond to influential parameters over its respective output in spite of non-linearity introduced by other parameters. Results are shown in Figure 5.8a and 5.8b for heterogeneous and homogeneous errors respectively.

Results for heterogeneous error show that the *swarm size* parameter is only positively highly correlated (absolute correlation coefficient greater than 0.7) with the *total items collected* output (Figure 5.8). This means that as the *swarm size* increases, the *total items collected* output increases too and this is because there are more robots collecting items in the arena.

The *d* parameter is negatively highly correlated with the *total items collected* and *collection ratio*. As *d* decreases, it takes less time to transport the items from the items source to the home area. In addition, since the distance the robots are travelling is smaller the probability of finding items increases.

Results for homogeneous error show that the *swarm size* parameter is not only highly correlated with the *total items collected* output but also with the *explore ratio* and *collection ratio* (Figure 5.8). As the *swarm size* increases, time spent in the *explore* state increases because robots spend more time avoiding each other. As consequence robots travel more when transporting items. Henceforth, the error increases and the probability of finding items decreases. This is reflected in the *collection ratio* output which it has a highly negative correlation with the *swarm size* parameter.

In a similar way, *d* is not highly correlated with *total items collected* and *collection ratio* but also with the *explore ratio* for both error types. As *d* increases, the probability of finding items decreases, as explained before, therefore, robots spend more time exploring than transporting items because they are getting lost more often.

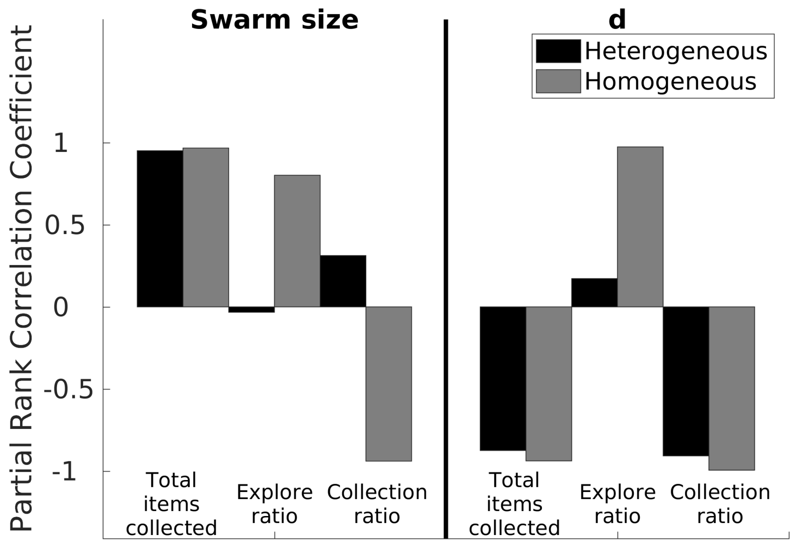


Figure 5.8: LHA for NPS with heterogeneous and homogeneous errors. Correlations change for each error type except for the *total items collected* for both parameters (*swarm size* and *d*) and collection ratio for *d* parameter.

Even though robots with homogeneous and heterogeneous errors are performing the same strategy, NPS, with the same settings, results from LHA differ from each other. The main reason is the individual errors in each robot for the heterogeneous error affect the performance of the swarm performance as a whole in different ways and this is explained next.

Total items collected is very similar for homogeneous and heterogeneous error as shown in Figure 5.9 a and 5.9 b. However, as for the *explore ratio* the correlation coefficient is very different from each other, -0.03 for heterogeneous error and 0.8 for homogeneous error. As shown in Figure 5.9c, with heterogeneous error model there are fluctuations across the *swarm size* parameter space and this is because of the individual errors considered. For instance, between *swarm size* 2 and 3 robots (robots A, B and C), the *explore ratio* ranges from 0.4 to 0.8. This is because this group of robots are characterized by their small error compared to other robots in the swarm, as shown in the error models in the previous section, therefore these robots spend more time transporting items than exploring. However, this does not happen with homogeneous error as shown in Figure 5.9d, where the *explore ratio* increases steadily without any oscillations present.

The *collection ratio* output for heterogeneous error also has fluctuations as shown in Figure 5.9e. In the regions where high error robots are introduced the *collection ratio* drops and increases again when low error robots are introduced (i.e. range between 5 and 7 robots). However, with homogeneous error the *collection ratio* decreases steadily (Figure 5.9f) and there are no fluctuations because no high error robots are introduced at any point because all the robots have the same error.

As shown, results are different from LHA between heterogeneous and homogeneous error. This is because of the fluctuations introduce between high and low error robots that affect the correlation coefficients. Complementary LHA scatter plots can be found in the Appendix B.

Extended Fourier Amplitude Sensitive Testing

The extended Fourier Amplitude Sensitive Testing (eFast) technique partitions the output variance caused by a parameter perturbation between the input parameters. This provides a strong indication of the influence of each parameter over each output. In addition, a 'dummy' parameter is introduced, which does not have impact on the emulation and helps to measure if the contribution by a single parameter is statistical significant different compared with the dummy parameter used as baseline. *Si* represents the fraction of variance accounted by that parameter and *STi* represents the fraction of variance accounted by that parameter and any higher-order or non-linear effects between this parameter and others.

The variance for the *total items collected* output is mainly caused by the *swarm size* parameter but *d* also contributes to it with a less degree (Figures 5.10a and 5.10b). The variance to *collection ratio* is caused by only *d* (Figures 5.10e and 5.10f). There is no noticeable difference between the homogeneous and heterogeneous error for these outputs, *total items collected* and *collection ratio*, however, it is different for the *explore ratio* output.

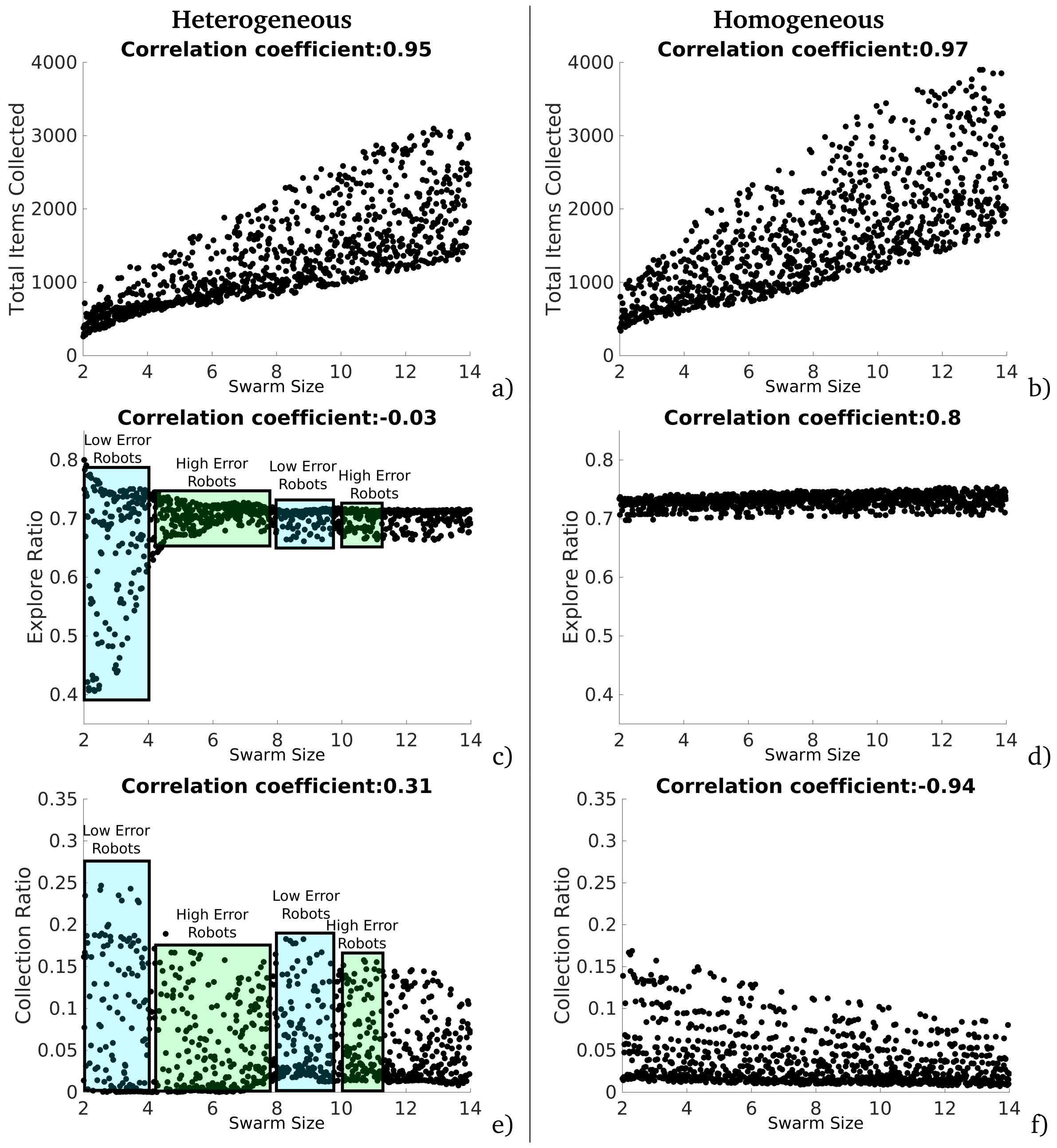


Figure 5.9: LHA for the *swarm size* parameter for NPS with heterogeneous (left column) and homogeneous (right column) errors. Robot individual error models introduce fluctuations with heterogeneous error as shown with the *explore ratio* (c) and *collection ratio* (e). However,this doesn't happen with homogeneous error (b, d and f) because all the robots share the same error parameters.

As for the *explore ratio*, the *Si* values are small which mean that contribution by the parameter itself is low. On the other hand, there are high values for *STi* and this represents that there are non-linear effects contribution to the *explore ratio* for both parameters (*swarm size* and *d*). This is because of the non-linearity introduced by the fluctuations of considering heterogeneous errors for the robots (Figure 5.10c). However, this cannot be seen with homogeneous error where variance to the *explore ratio* output is caused by *d* (Figure 5.10d).

This contrast of results in the *explore ratio* can be linked to LHA where this output in specific suffers from big changes from correlation coefficients (Figure 5.9). Therefore, the eFast technique identifies this non-linearity in the heterogeneous output.

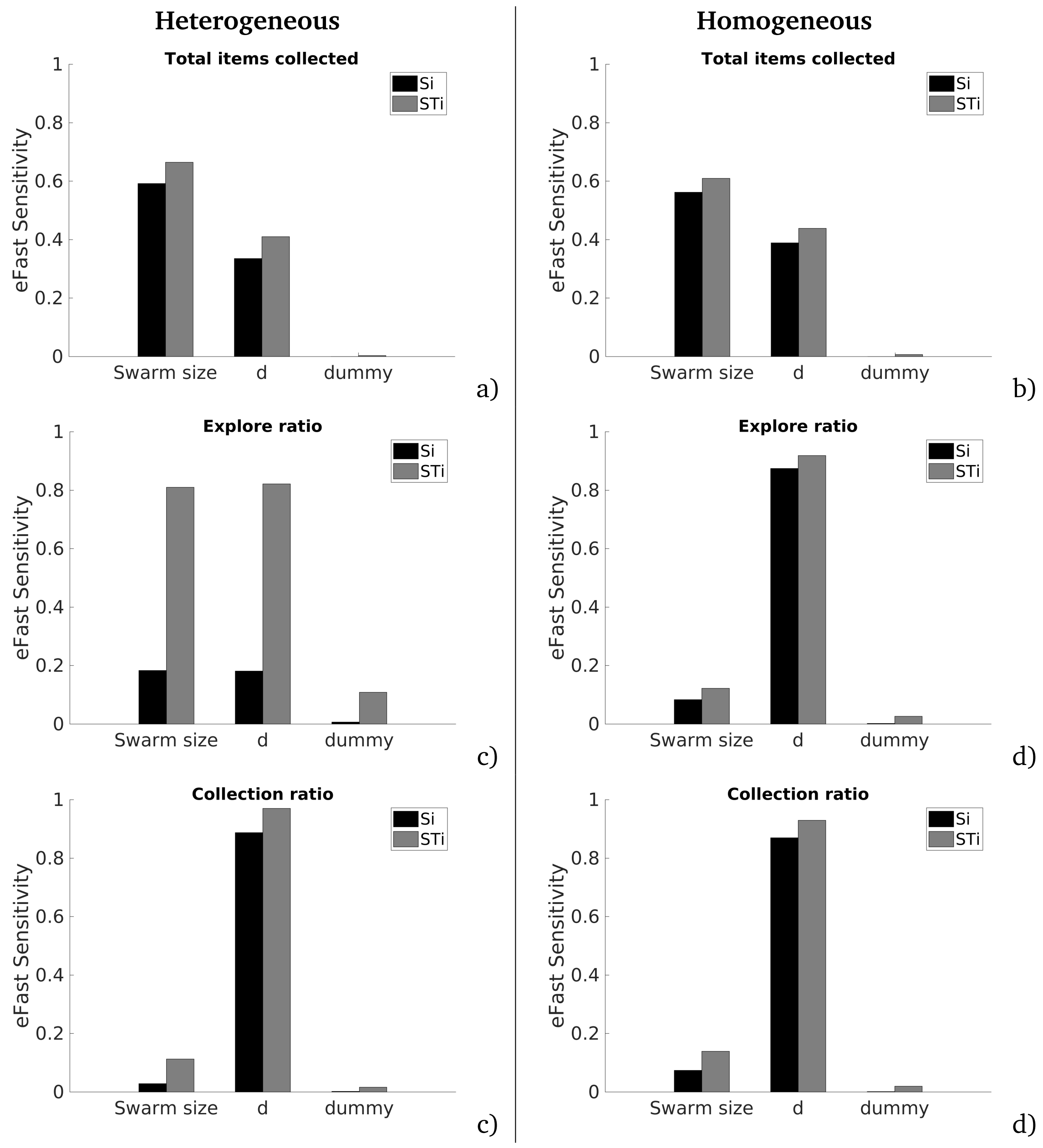


Figure 5.10: eFast results for NPS with heterogeneous (left column) and homogeneous error (right column). Results differ for the *explore ratio* output between heterogeneous (c) and homogeneous (d) errors.

Efficient Approximate Bayesian Computation

The efficient Approximate Bayesian Computation (easyABC) provides a parameter distribution layout that gives rise up a specific expected behaviour. In other other words, easyABC provides a density measure where the values for a specific parameter lay that exhibit a specific behaviour.

Results for the *total items collected* output are shown here but for the rest of the outputs can be found in the Appendix. The reason why the rest of the outputs are excluded from this section is because these results show similar distribution to the normal distribution because there are not big changes between the summary statistics and the range of values provided by each parameter.

Figure 5.11 shows the results for the *total items collected* output with respect each input (*swarm size* and *d*) for heterogeneous and homogeneous errors. The summary statistics selected is 3000 items because this is the maximum number of items collected in LHA. This time there is no major difference between heterogeneous and homogeneous results except that results with homogeneous error show better defined distribution. This is because of the non-linearities just described before. In order to get that number of items the *swarm size* parameter is negatively skewed meaning that in order to have a high number of items it is necessary to have a big group of robots (Figures 5.11a and 5.11b). As for *d*, the distribution is positively skewed which means that a small *d* provides high item collection (Figures 5.11c and 5.11d).

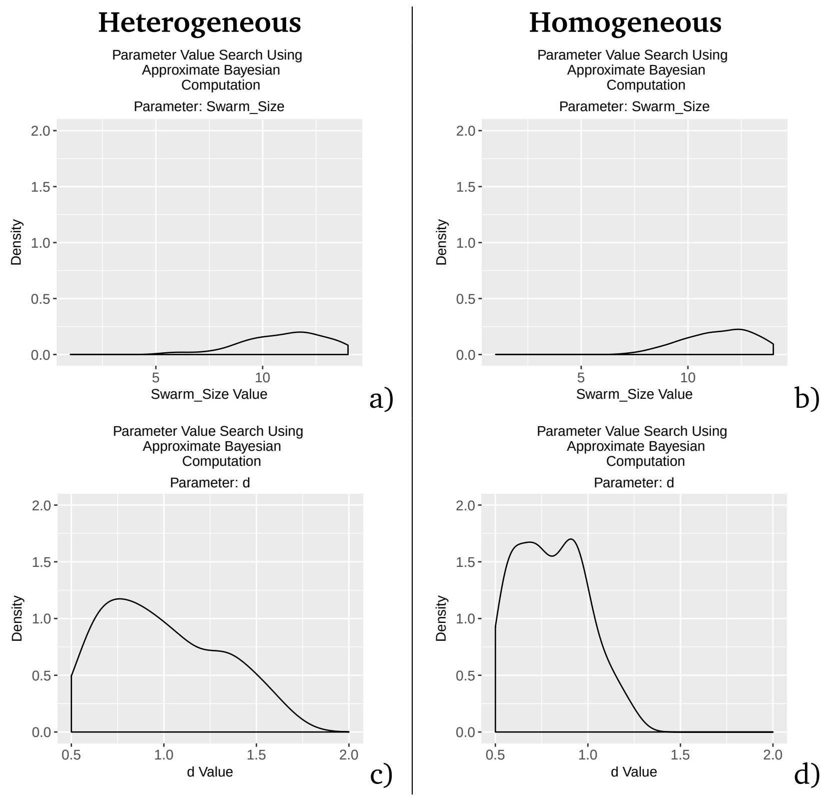


Figure 5.11: easyABC *total items collected* results for NPS with heterogeneous (left column) and homogeneous (right column) error for the *swarm size* (a and b) and *d* parameters (c and d).

Non-dominated Sorting Genetic Algorithm II

Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a technique that explores the entire parameters space in order to maximize and/or minimize multiple outputs. Once, all the solutions population converge (or meet the specific convergence criteria) this set of solutions is referred as pareto front.

NSGA-II is used here to find the pareto front for the best values for *swarm size* and *d* parameters in order to maximize *total items collected* and *collection ratio*, and minimize *explore ratio* as shown in Figure 5.12. The pareto front is discontinuous for the heterogeneous error (Figures 5.12a, 5.12c and 5.12e), as the emulation captures the heterogeneity in the errors. For instance, for small swarm sizes, from 2 to 3 robots (robots A, B and C), the error is small which means that the robots can travel longer lengths. This minimizes the *explore ratio* and they spend the least amount of time avoiding each other. Robots between the swarm size 2 and 4 robots with *d* of 0.5 maximize the collection ratio. This is because the robots are travelling the shorter distance between home area and items source, therefore the accumulation is small. Between 5 and 7 (robots E, F and G), robots are characterized by their high degree of error. Therefore, these robots behave more like obstacles and they harm item collection. As consequence, the swarm is affected negatively. Lastly, between 8 and 14 (robots H, I, J, K, L, M and N), robots with low error are again introduced which contribute for a high *total items collected* output. Further work could potentially use this discontinuous pareto front in order to identify the threshold of number of high error robots that when overcome the swarm throughput is affected negatively. This would help to measure the degrees of robustness of the task.

Pareto front is continuous for the homogeneous error (Figures 5.12b, 5.12d and 5.12f). As all the robots contribute in the same way. If the *total items collected* was to be maximized the *swarm size* needs to be increased. However, if the *collection ratio* is maximized and the *explore ratio* minimized the size of the *swarm size* needs to be decreased. In order to optimize the three outputs it is necessary to have the smallest *d*.

Conclusion

From this analysis, it is possible to conclude that it is important to consider heterogeneous and homogeneous errors as the results from both can be misleading if they are considered separately. In addition, the implementation of different statistical techniques helps to provide a better understanding of the system. Finally, heterogeneity error introduces fluctuations and non-linearity to the system as shown with LHA and eFast. NSGA-II exploits the heterogeneity and finds new solutions that maximizes and minimize outputs. The strategies described in the following section decreased this by dividing the task into multiple components.

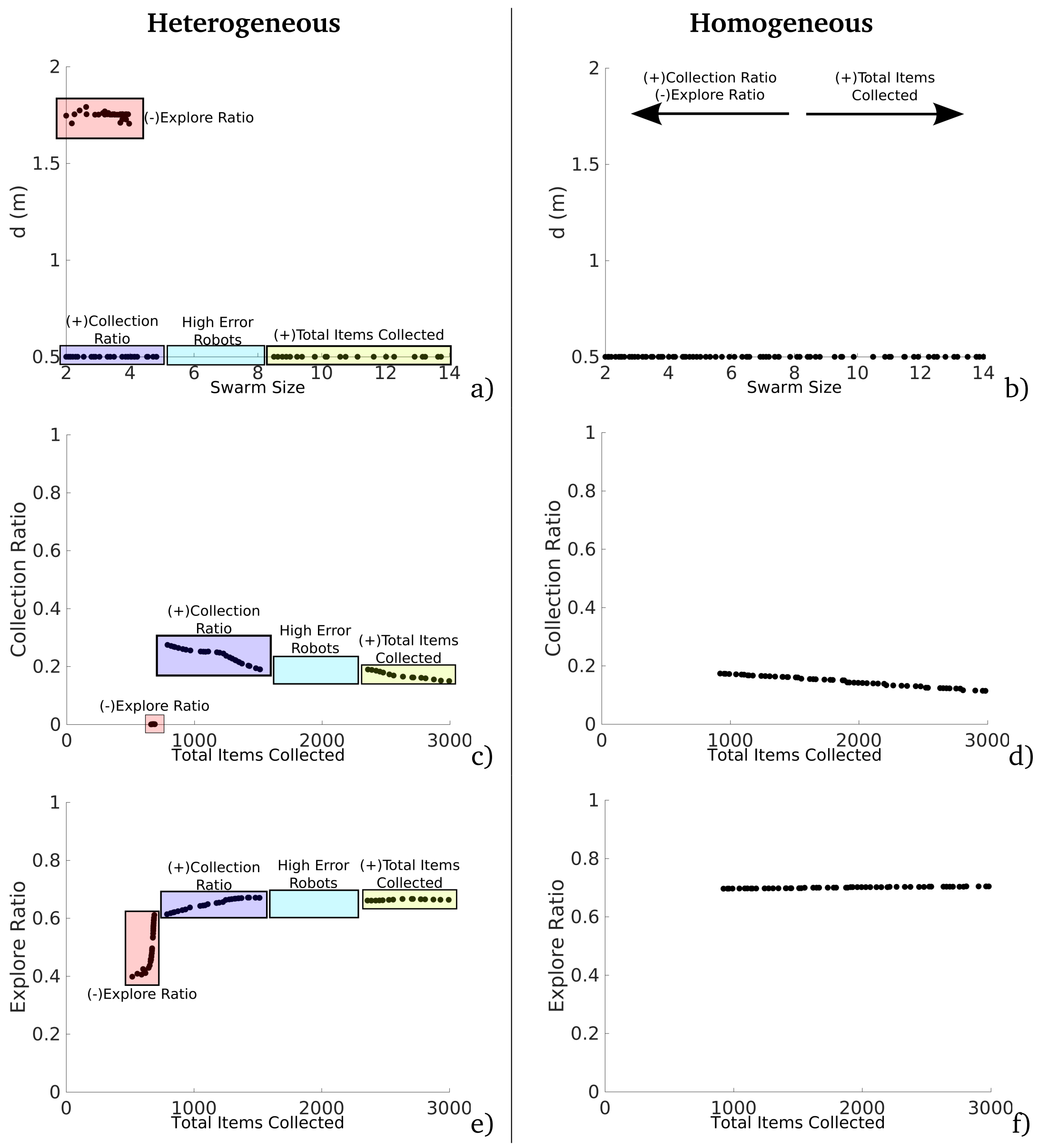


Figure 5.12: NSGA-II results for NPS with heterogeneous (left column) and homogeneous (right column) error. The two dimensional input is *swarm size-d* (a and b). Two dimensional outputs are shown as *total items collected-collection ratio* (c and d) and *total items collected-explore ratio* (e and f).

5.4.2 Static partitioning strategy

The Static Partitioning Strategy (SPS), introduced in Section 3.4, is where the robots travel predefined fixed distance towards the home area where they drop the items.

For the emulations the same two parameters used for NPS, *swarm size* and *d*, are again used but a new parameter is introduced which is the partition length (P).

P defines the amount of distance for the robots to travel towards the home area in the *go to nest* state after an item is found from the items source. After this distance is reached, the robot drops the item and then switches to the *go to source* state. In case of SPS, all the robots share the same P ($P_i = P$) and it does not change over time. For the experiments in this section the interval used is from 0.2 m to 2.0 m.

Latin-hypercube analysis

Results for heterogeneous and homogeneous error are shown in Figure 5.13 and similarities are described next. The *swarm size* parameter has only a direct high correlation with the *total items collected* output and this is due to the high redundancy of robots as seen with NPS too. In contrast to NPS, *swarm size* does not longer has a direct correlation with *explore ratio* and *collection ratio* because these outputs are now regulated by P. The P parameter has a direct positive correlation with the *explore ratio* and a negative correlation with the *collection ratio*. This means that as P increases the distance robots have to travel is longer which makes the dead-reckoning error to accumulate. As the probability of finding items decreases and the robots have to explore more. Lastly, *d* only has high correlation with the *total items collected* output because the amount of time transporting items decreases.

The results that differ between heterogeneous and homogeneous error are the correlation between *d* and *explore ratio* and *collection ratio*. In spite of this contrast between error types, the correlation values are low (absolute correlation coefficient smaller than 0.7). Fluctuations in SPS are smaller than with NPS because of the P parameter which decreases the error for all the robots increasing homogeneity across the swarm. This is shown in the next section with experiments in simulation and hardware.

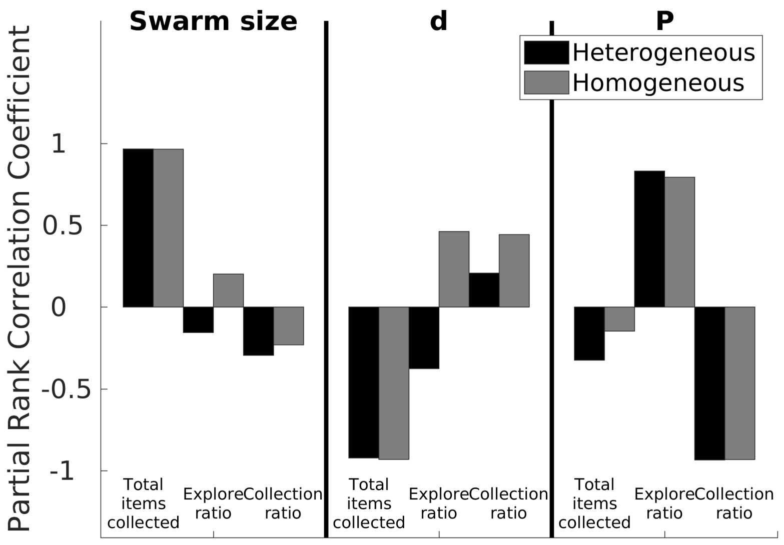


Figure 5.13: LHA for SPS with heterogeneous and homogeneous errors. Correlations are similar for both error types except of the parameter *d* with *explore ratio* and the *collection ratio*.

Even though, fluctuations have decreased, some outliers can be seen in the LHA scatter plots for *d* in Figures 5.14a and 5.14c. These outliers are caused when the swarm size is up to three robots (A, B and C) which they are characterized by their low error. Therefore, this specific swarm size provides the highest *collection ratio* with the lowest *explore ratio*. The outliers are not present with the homogeneous error (Figures 5.14b and 5.14d). However, as previously mentioned, the heterogeneity affects the SPS results in less degree than NPS.

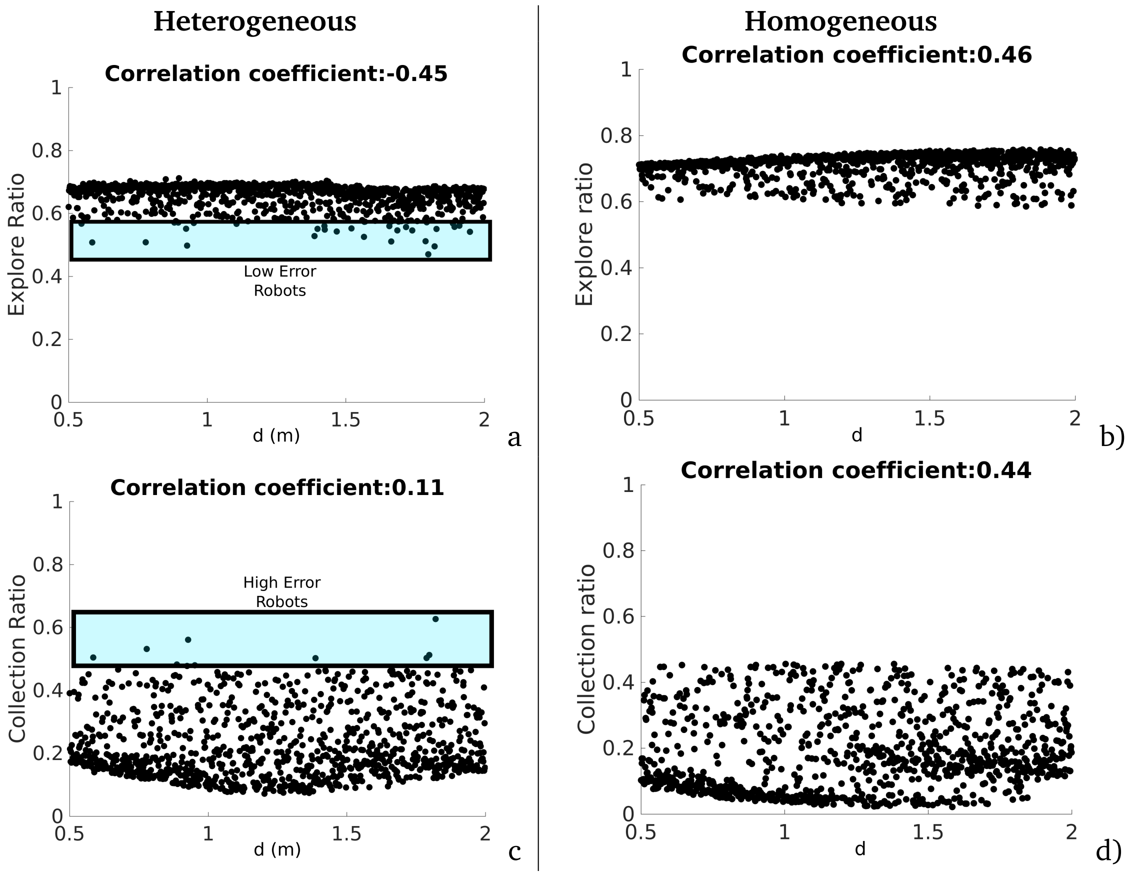


Figure 5.14: LHA for the *d* parameter for SPS with heterogeneous (left column) and homogeneous (right column) errors. Robot individual error models introduce fluctuations with heterogeneous error as shown with the *explore ratio* (a) and *collection ratio* (c). However,this doesn't happen with homogeneous error (b and d) because all the robots share the same error parameters.

Extended Fourier Amplitude Sensitive Testing

P plays a key role in the variation for *explore ratio* and *collection ratio* as shown in Figures 5.15c, 5.15d, 5.15e and 5.15f. As for the *total items collected* shown in Figures 5.15a and 5.15b, they are similar to the results with NPS where item collection is mainly affected by the *swarm size* but *d* still provides a contribution.

In contrast to NPS, fluctuations caused by heterogeneity are much smaller with SPS . STi for *swarm size* and *d* is smaller than 0.2 for SPS (Figure 5.15c) where for NPS is greater than 0.8

(Figure 5.10c).

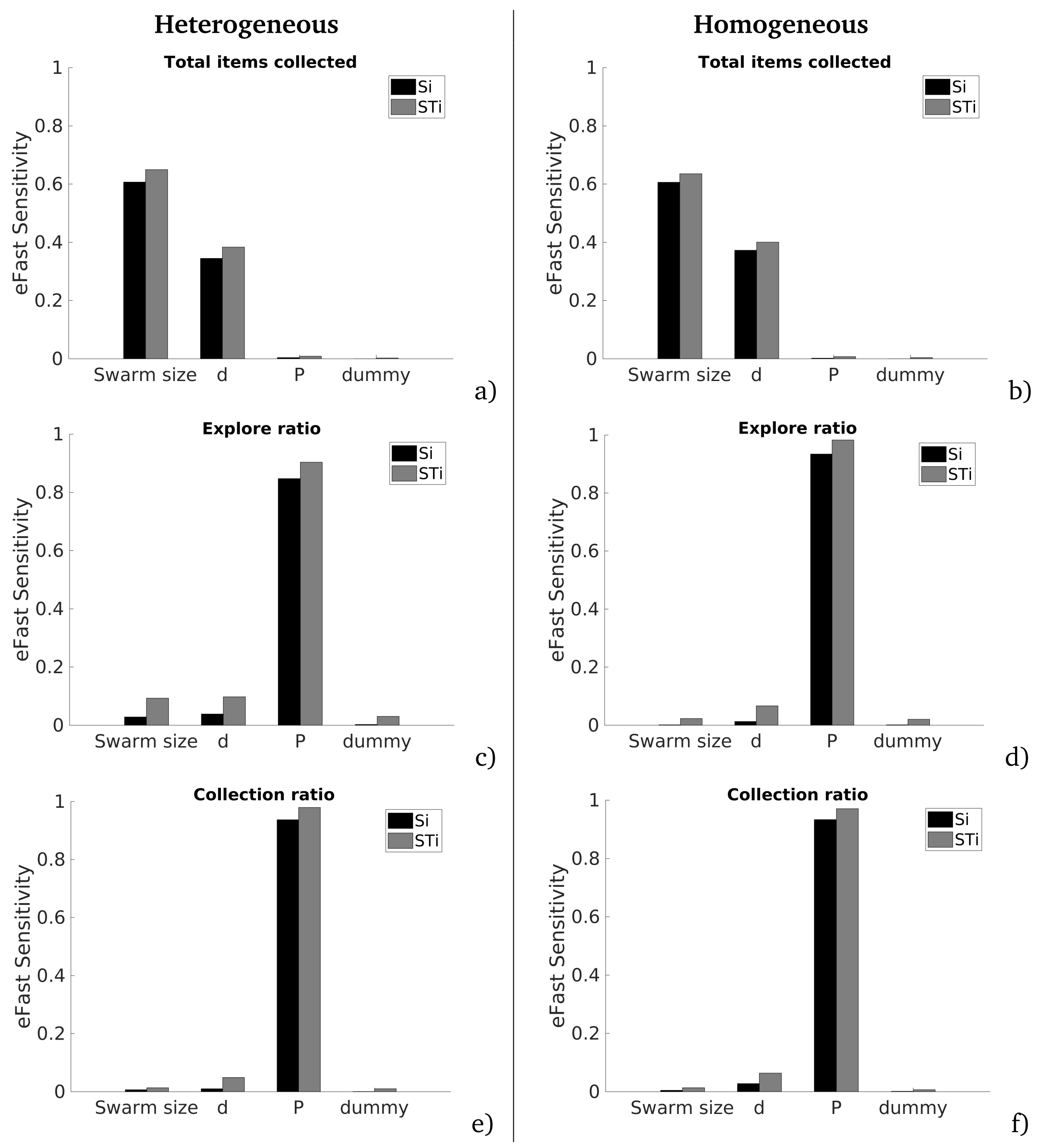


Figure 5.15: eFast results for SPS with heterogeneous (left column) and homogeneous error (right column). Results are very similar between heterogeneous (c) and homogeneous (d) errors.

Efficient Approximate Bayesian Computation

The results for the *total items collected* output (Figures 5.16 a, b, c and d) are similar to the results from NPS. Density is negatively skewed for the *swarm size* and positively for *d*. In addition, density is greater with homogeneous error.

The results differ between heterogeneous and homogeneous error for the *d* parameter. Results for the *explore ratio* is negatively skewed for the heterogeneous error (Figure 5.16f) with a maximum density of 1.2 whereas density appears to have a normal distribution for the homogeneous error (Figure 5.16e) with a lower density close to 0.825. The *collection ratio* output experiences a similar contrast between the heterogeneous and homogeneous error. Density is positively skewed for the heterogeneous error with a maximum density close to 1.2 (Figure 5.16h). These results are consistent with the results provided by the LHA (Figure 5.9) which means that fluctuations introduced by the heterogeneous error are still present.

Non-dominated Sorting Genetic Algorithm II

Results for SPS for heterogeneous error are shown in the left column in Figure 5.17. In order to maximize the item collection, it is necessary to have all the 14 robots in the smallest arena ($d = 0.5$) and travelling the maximum P. Big values for P guarantee that there are no partitions (Figure 5.17a and 5.17c). However, *explore ratio* and *collection ratio* are affected negatively by this because of the high number of robots in the environment, causing them to spend more time avoiding each other than transporting items (Figure 5.17e and 5.17g). On the other hand, a small swarm size (2 robots) and a small P, provides the highest *collection ratio* and the lowest *explore ratio*. Nevertheless, a high *d* provides more space for the robots to transport items which decreases the *explore ratio* and a small *d* increases the probability for the robots to find more items, and as consequence increasing the *collection ratio*. Heterogeneity in the individual errors create two gaps where high error robots are introduced. In addition, since robots A, B, and C have low error, these robots maximize and minimize *collection ratio* and *explore ratio* respectively.

In order to maximize the item collection it is necessary to have all the 14 robots in the arena in a similar way than with the heterogeneous error (Figures 5.17b and 5.17d). In contrast to the heterogeneous error where NSGA-II exploits individual errors, there is no point having a small swarm size because all the robots have the same performance. Optimization for *explore ratio* and the *collection ratio* is mainly defined by *d* and P. As *d* increases, *explore ratio* decreases and *collection ratio* increases, because they spend less time avoiding each other and error decreases because robots are travelling less when transporting items (Figures 5.17f and 5.17h). Lastly, as P decreases the *explore ratio* increases and the *collection ratio* increases because robots are travelling shorter lengths. Therefore, the probability of finding items increases the accumulation of dead-reckoning error is smaller.

Conclusion

In conclusion, the P parameter helps to increase the *collection rate* and to decrease the time spent in the *explore* state, hereby increasing the time spent transporting items. However, as it was shown in the previous section, results from heterogeneous error are still affected by

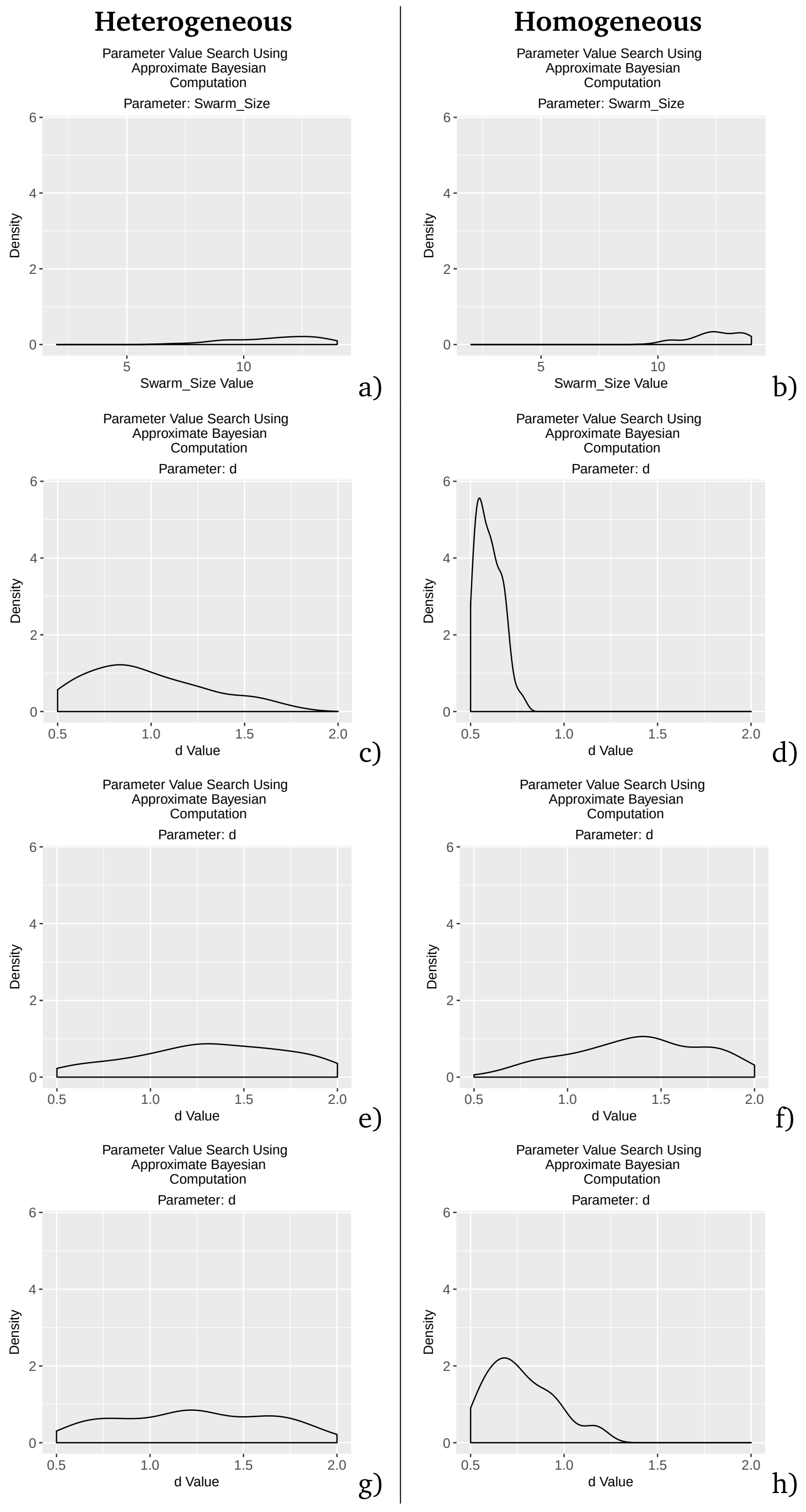


Figure 5.16: easyABC results for SPS with heterogeneous (left column) and homogeneous (right column) error for the *total items collected* output with the *swarm size* (a and b) and *d* parameters (c and d), and for the *explore ratio* (e and f) and *collection ratio* (g and h) for the *d* parameter.

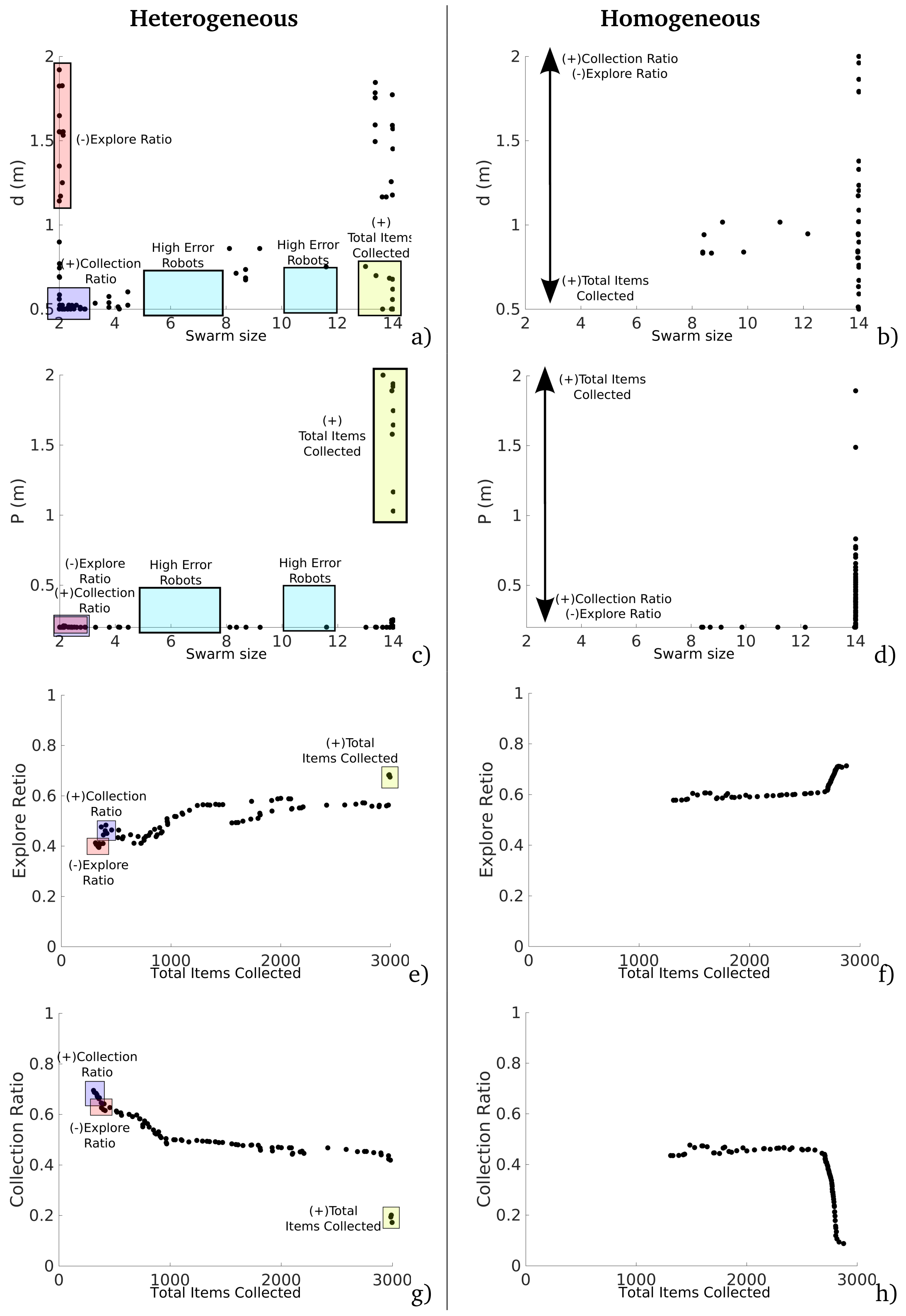


Figure 5.17: NSGA-II results for SPS with heterogeneous (left column) and homogeneous (right column) error. Two dimensional inputs are shown as swarm size-*d* (a and b) and swarm size-P (c and d). Two dimensional outputs are shown as *total items collected-explore ratio* (e and f) and *total items collected-collection* ratio (g and h).

fluctuations introduced by the individual errors. The pareto front is still discontinuous for the heterogeneous error due to NSGA-II exploiting the low error robots. In the the next part, DPS addresses this issues with its learning mechanism.

5.4.3 Dynamic partitioning strategy

The Dynamic Partitioning Strategy (DPS), introduced in Chapter 3.4, is the strategy where robots change their individual partition length (P_i) according to a penalty and reward system. Instead of using a single fixed partition P as with the previous strategy, SPS, P_i changes with the parameter α.

α regulates the amount of penalty and reward to P_i . As α increases the robot i gets more penalized than rewarded and as α decreases the robot is more rewarded than penalized. The interval used for the experiments shown in this section is [0,1]. Initial P_i is randomly selected from the same interval as *d*.

Latin-hypercube analysis

Results between heterogeneous and homogeneous error are very similar for all the correlations except for the correlation between the *swarm size* parameter and the *explore ratio* and the *collection ratio* outputs as shown in Figure 5.18. This discrepancy is again produced by the heterogeneity within the individual error. However, the correlation is kept low (absolute correlation coefficient lower than 0.7).

The *total items collected* output is once again mainly correlated to the *swarm size* and *d* parameters for the reasons previously explained in the previous strategies. However, α has a negative correlation with *explore ratio* and a positive correlation with *collection ratio*, This means that as α increases the robots are exploring less and finding items more often because all the robots are adjusting their P_i to achieve this.

The discrepancy mentioned before for the *swarm size* parameter is shown in Figure 5.19. Figures 5.19a and 5.19c show that for a small swarm size (less than 4 robots) the *collection ratio* decreases and the *explore ratio* increases whereas as for the rest of the swarm this correlation pattern can not be seen, this could be because high error robots are compensating this for the heterogeneous error. Whereas in Figures 5.19b and 5.19d, the *explore ratio* and the *collection* ratio decreases and increases steadily respectively for the homogeneous error. As the swarm size increases, there is higher supply of robots to create a chain successfully between the home area and the items source.

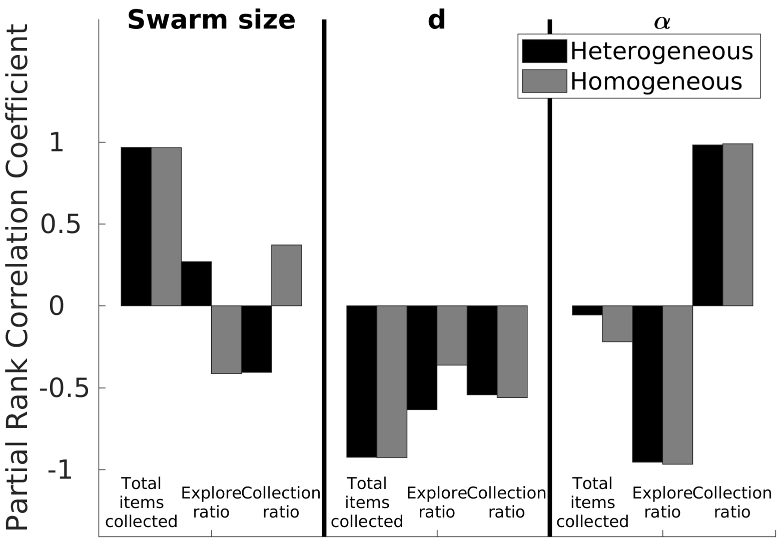


Figure 5.18: LHA for DPS with heterogeneous (a) and homogeneous (b) error. Results are similar between heterogeneous and homogeneous error except for correlation between the *swarm size* parameter and the *explore ratio* and *collection ratio*.

Extended Fourier Amplitude Sensitive Testing

The results from this technique are very similar to the results for the SPS strategy and there is little difference between heterogeneous and homogeneous error (Figure 5.20). The *total items collected* is mainly varied by the *swarm size* and *d* in the second place (Figures 5.20a and 5.20b). α causes mainly variations to the *explore ratio* and the *collection ratio* (Figures 5.20c, 5.20d, 5.20e and 5.20f).

Efficient Approximate Bayesian Computation

Density is negatively skewed for the *swarm size* and positively for *d*. In addition, density is greater with homogeneous error. Results from the easyABC technique for the *total items collected* output (Figures 5.21 a, b, c and d) are similar to the results from SPS.

The results differ between heterogeneous and homogeneous error for the *collection ratio* output. Results for the *swarm size* parameter is negatively skewed for the homogeneous error (Figure 5.21f) with a maximum density of 0.1 whereas density appears to have a normal distribution for the heterogeneous error (Figure 5.21e) with a lower density close to 0.15. The *collection ratio* output experiences a similar contrast between the heterogeneous and homogeneous error. Density is positively skewed for the homogeneous error with a maximum density close to 1.5 (Figure 5.21h). These results are consistent with the results provided by the LHA (Figure 5.21) which means that fluctuations introduced by the heterogeneous error are still present.

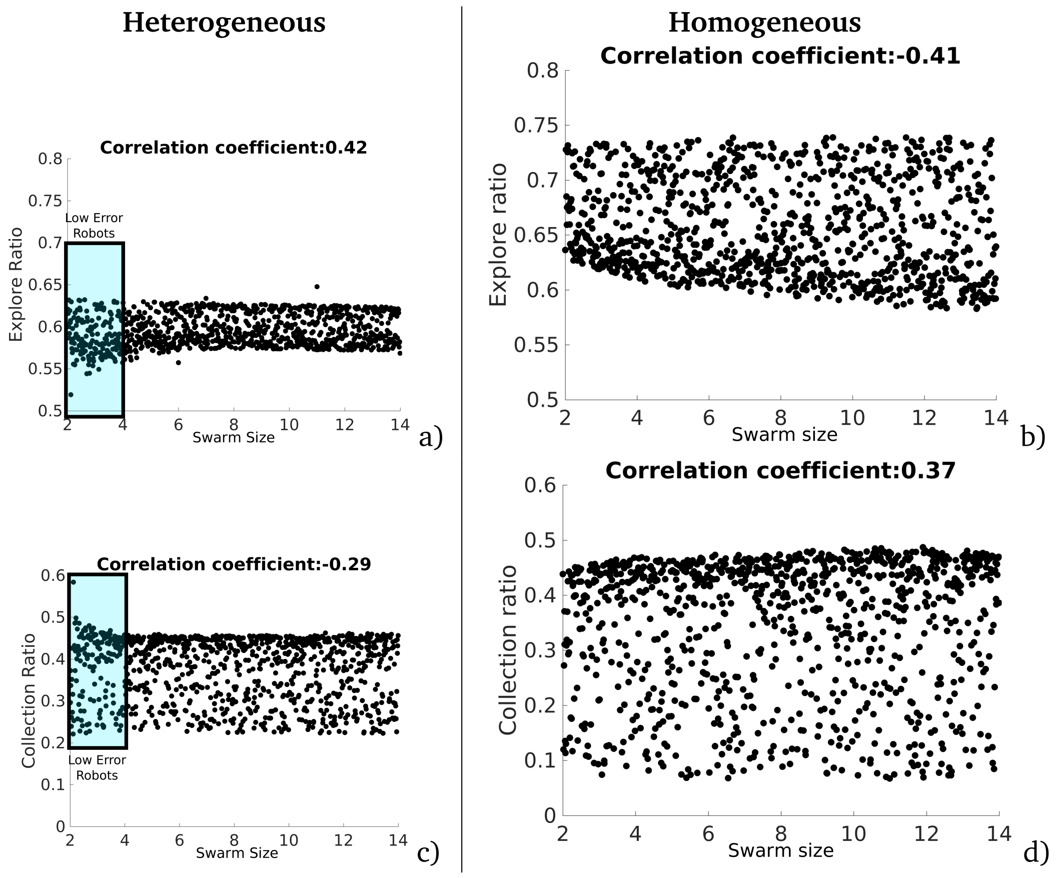


Figure 5.19: LHA for the *d* parameter for DPS with heterogeneous (left column) and homogeneous (right column) errors. Fluctuations introduce by low error robots affect estimation of the correlation coefficient.

Non-dominated Sorting Genetic Algorithm II

Similar as with SPS, a big swarm size provides the best item collection and α regulates the *explore ratio* and ratio *collection* where as α increases the *collection ratio* increases and the *explore ratio* decreases (Figures 5.22a and 5.22c). In addition, in a similar way, NSGA-II for the heterogeneous error exploits the small swarm size in order to achieve the best *explore ratio* and *collection ratio* (Figures 5.22e and 5.22g).

The pareto front for DPS is smother than SPS and discontinuities or gaps are smaller (Figures 5.22e and 5.22g). This is because DPS has a better regulation over the individual errors creating a closer to homogeneous results across the swarm. The main reason is because each robot i is learning its own P_i for the given α (more details in the next section). This can be also seen in Figures 5.22a and 5.22c where there are only two batches of solutions, the ones closer to swarm size of 2 robots and the ones closer to the swarm size 14 which means that fluctuations by high and low error robots between the interval 4 and 10 are ignored and have no significant contributions to the results. This could be seen in Figure 5.19 where there are no fluctuations in this interval where SPS does have solutions in the swarm size interval 8

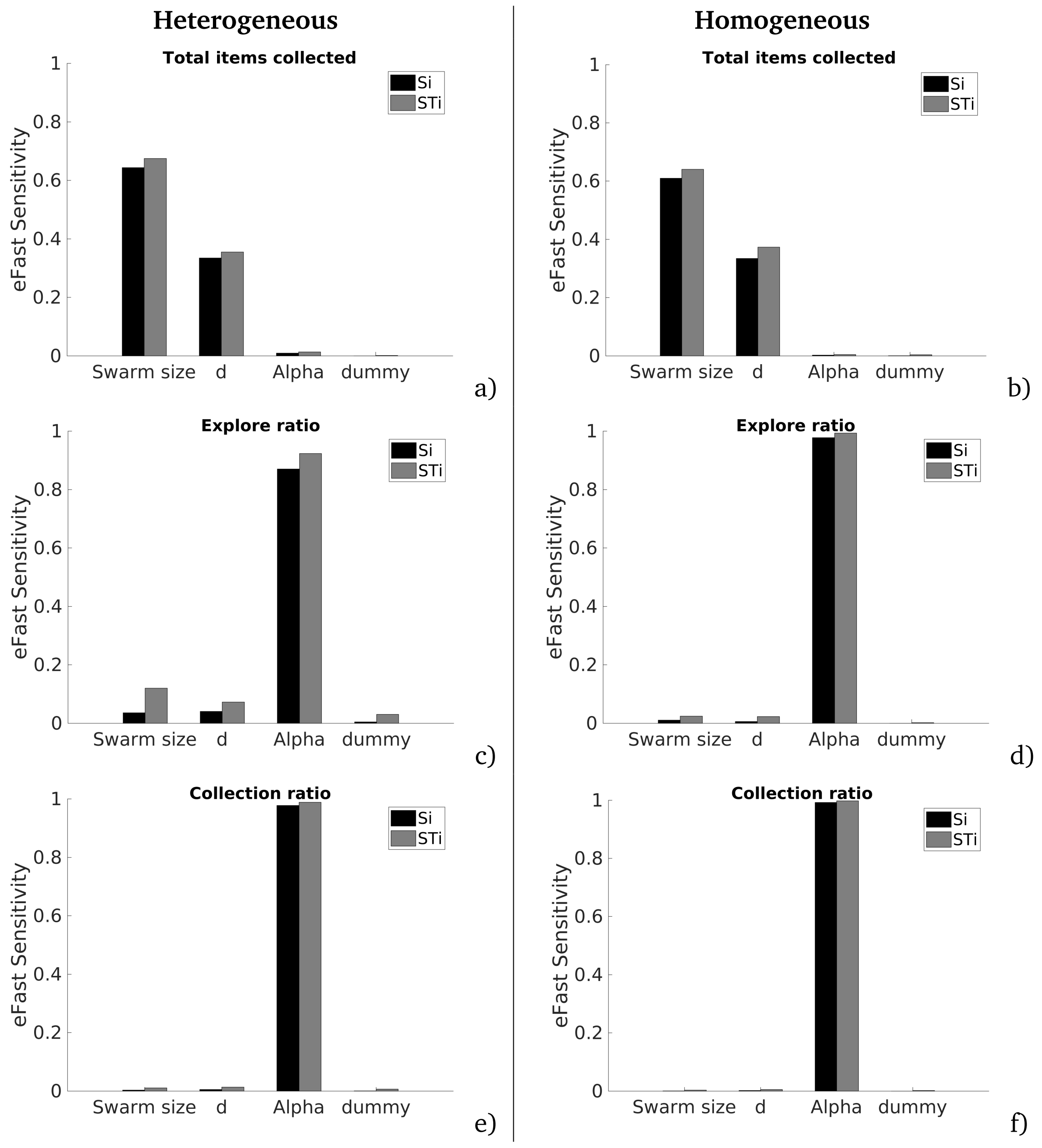


Figure 5.20: eFast results for DPS with heterogeneous (left column) and homogeneous error (right column).

to 10 robots that are characterized by their low error (Figure 5.17).

Results for the homogeneous error show that in a similar way as with SPS a big swarm provides the best item collection (Figures 5.22b and 5.22d). As α increases the *collection ratio* increases and the *explore ratio* decreases. The pareto front for the set of solutions is continuous and smooth (Figures 5.22f and 5.22h). The collection ratio for the homogeneous error for DPS is greater than with SPS.

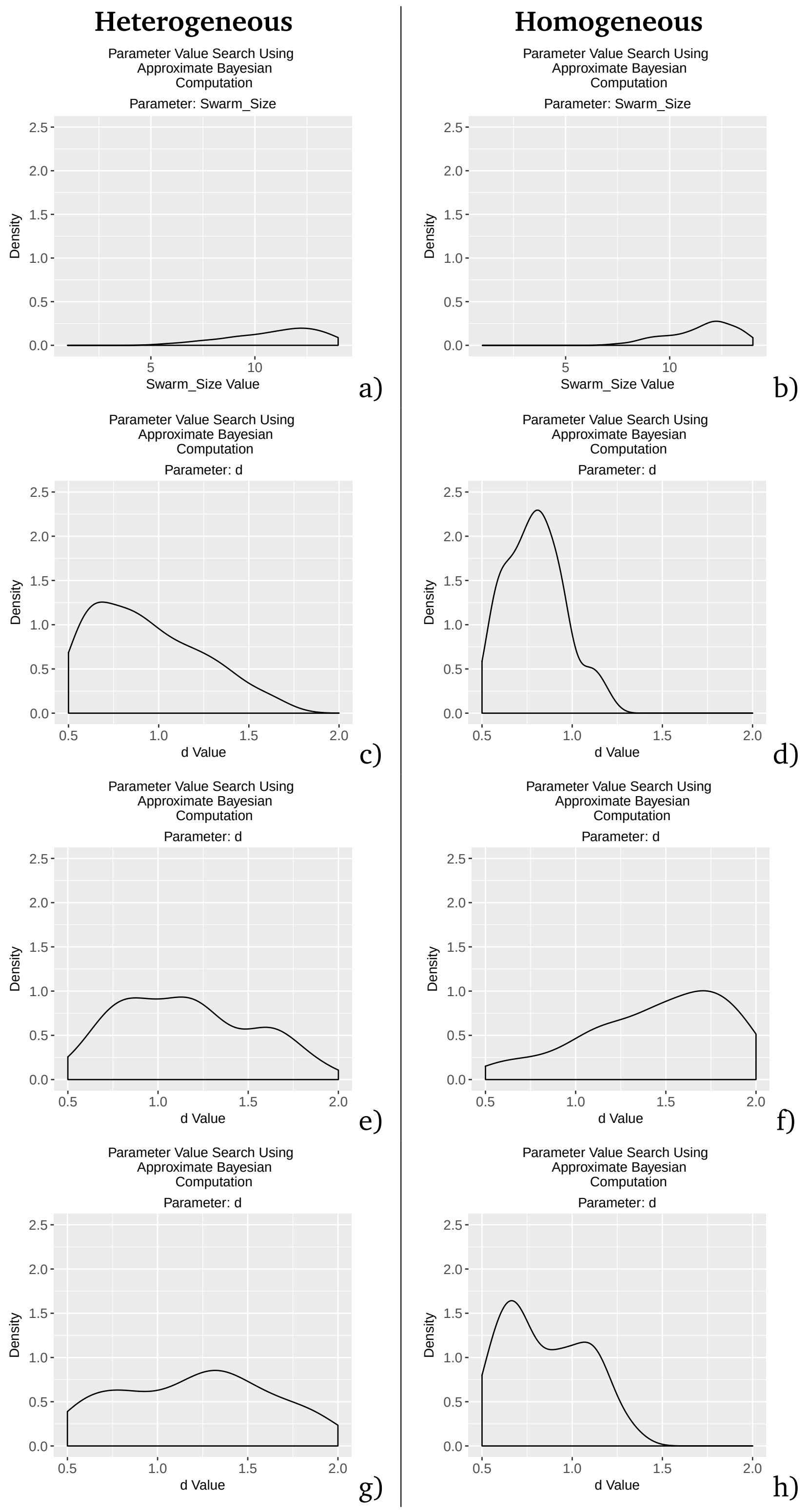


Figure 5.21: easyABC results for DPS with heterogeneous (left column) and homogeneous (right column) error for the *total items collected* output with the *swarm size* (a and b) and *d* parameters (c and d), and for the *collection ratio* (e and f) and *collection ratio* (g and h) for the *d* parameter.

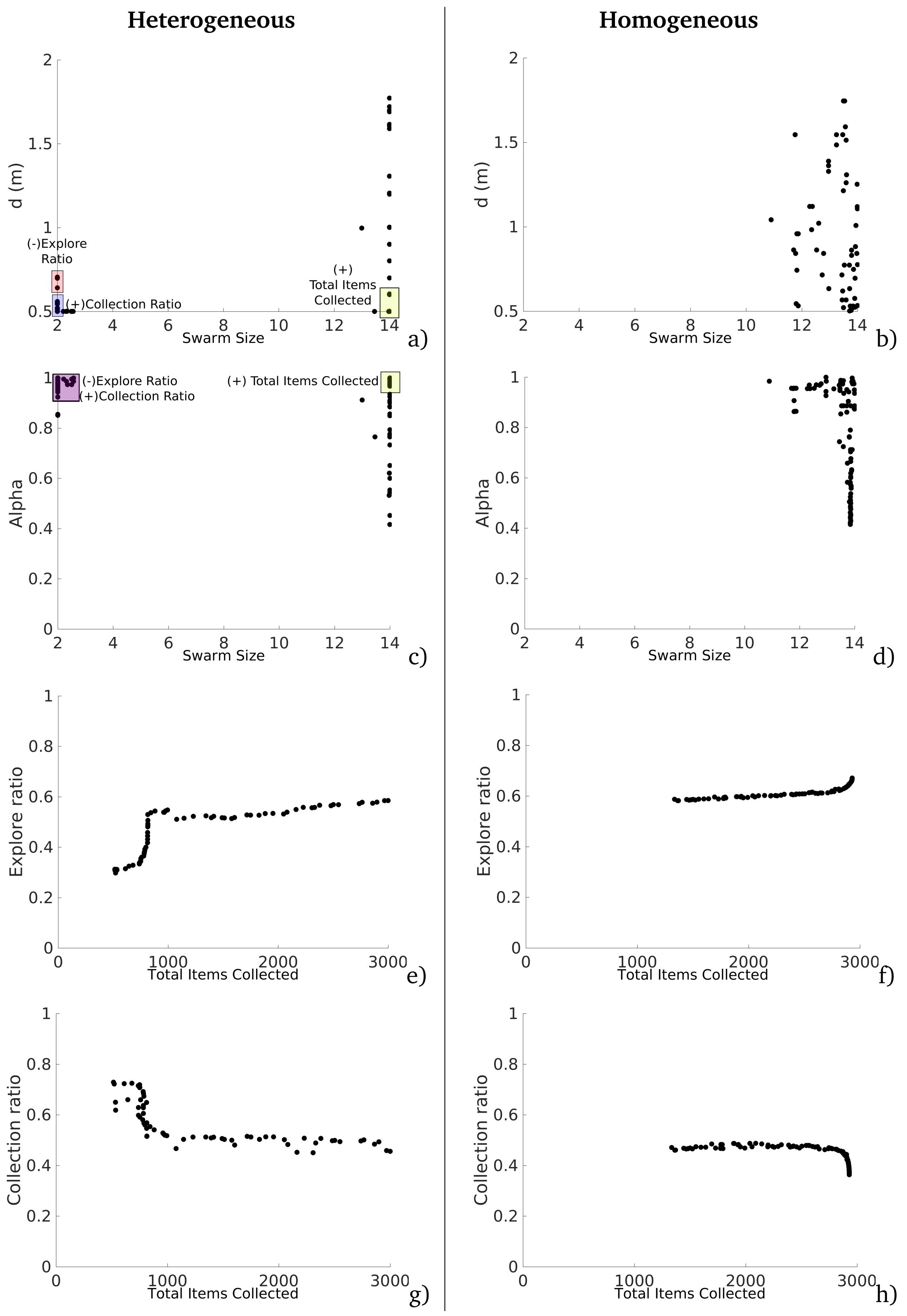


Figure 5.22: NSGA-II results for DPS with heterogeneous (left column) and homogeneous (right column) error. Two dimensional inputs are shown as *swarm size-d* (a and b) and *swarm size-alpha* (c and d). Two dimensional outputs are shown as *total items collected-explore ratio* (e and f) and *total items collected-collection* ratio (g and h).

Conclusion

In conclusion, α helps to regulate the *collection ratio* in order to optimize the *explore ratio*. The *total items collected* output is provided mainly by the *swarm size* and *d*. In this way, it is possible to find a set of parameters the provide a good trade-off between the *explore ratio* and the throughput of items according to the needs of the user. Finally, fluctuations caused by individual error models are reduced and a continuous pareto front for the output is generated for the heterogeneous error because of the individual convergence of P_i for each robot. More information about P_i convergence can be found in the experiments from simulation and hardware in the next section.

5.4.4 Cost partitioning strategy

The cost-based approach partitioning strategy (CPS), introduced in Chapter 3.4, consists of robots dividing *d* into a set of smaller discrete lengths. Every time a robot grips an item, the robot chooses randomly a partition from the set and it measures the time that it takes to find another item again. The partition length that has the lowest time (cost) has a higher probability to select that partition.

An $\epsilon - greedy$ algorithm selects the next partition length from the set. ϵ defines the amount of exploration and exploitation. On one hand, a small value of epsilon provides high exploitation which means that a robot selects the partition with the lowest cost (least amount of time spent in the partition) but the swarm is less resilient to changes in the environment or in the robot itself. On the other hand, a value closer to 1 provides high exploration which means that robots have a higher probability of selecting partition lengths with higher costs and with this the swarm is more flexible against changes because of its capability of finding new solutions.

The memory factor defines the influence of previous time estimations on the current calculation. A value closer to 0 gives priority to the actual estimation and a value closer to 1 priority is provided to previous estimations.

Latin-hypercube analysis

Results between heterogeneous and homogeneous error share some strong similarities where the *swarm size* has a high direct correlation with the *total items collected* output, and *d* is highly correlated to the *total items collected* and the *collection ratio* (Figure 5.23). These correlations are similar to the results with NPS. However, results differ mainly for the correlation between *swarm size* and *collection ratio*, and *d* and *explore ratio*.

Figure 5.24 shows the scatter plots for the main discrepancies between error types. Outliers can be seen for the heterogeneous error. For instance, between the *swarm size* 2 and 3 robots, the *collection ratio* correlation coefficient is negative mainly because of these outliers that

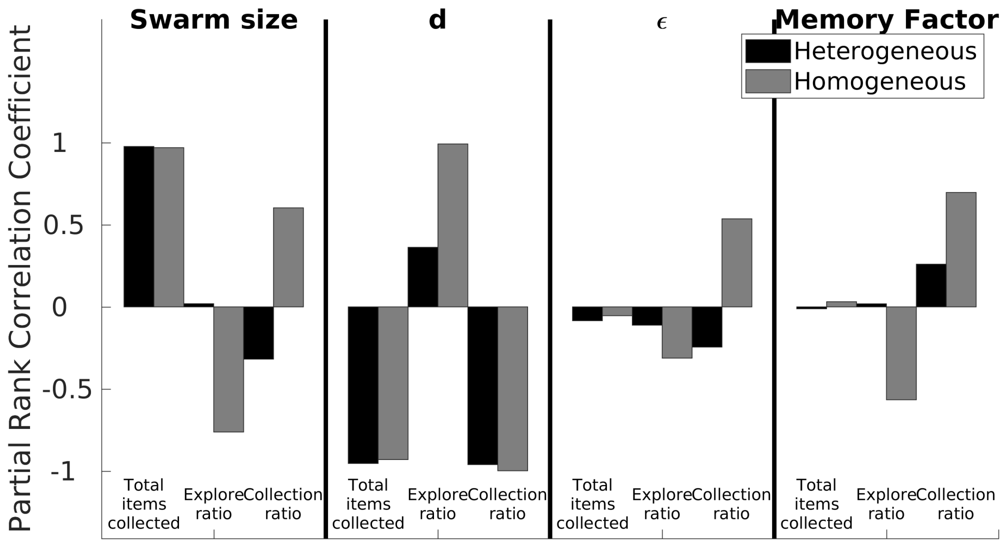


Figure 5.23: LHA for CPS with heterogeneous (a) and homogeneous (b) error. Results are similar between *swarm size* and *total items collected*, and between *d* and *total items collected* and *collection ratio*. Correlations are mainly different between *d* and the *explore ratio*.

decrease (Figure 5.24a). Outliers can be found across the scatter plot for *d* as shown in Figure 5.24c. As for the heterogeneous error, as *d* increases the *explore ratio* increases with it with a strong correlation coefficient. This is consist with NPS which means that CPS for homogeneous error might not be contributing in increasing the performance of the swarm by decreasing the *explore ratio* and the *collection ratio*.

Extended Fourier Amplitude Sensitive Testing

Results between both error types are similar for the *total items collected* and the *collection ratio* (Figures 5.25a, 5.25b, 5.25e and 5.25f). High STi values are found for the *explore ratio* output (Figures 5.25c and 5.25d). In contrast to NPS, STi values remain high for the homogeneous error type which could lead to speculate that not only fluctuations by individual errors introduce non-linearity to the the results but also, CPS is sensible to the homogeneous high error distribution (Figure 5.4).

Efficient Approximate Bayesian Computation

Results for the heterogeneous error (left column) have a structure similar to a normal distribution whereas for the homogeneous error (right column) for *total items collected* and *explore ratio* for both parameters (*swarm size* and *d*) are skewed. This is consistent with the results from NPS where *swarm size* and *d* play a key role for *the total items collected* and the *explore ratio*.

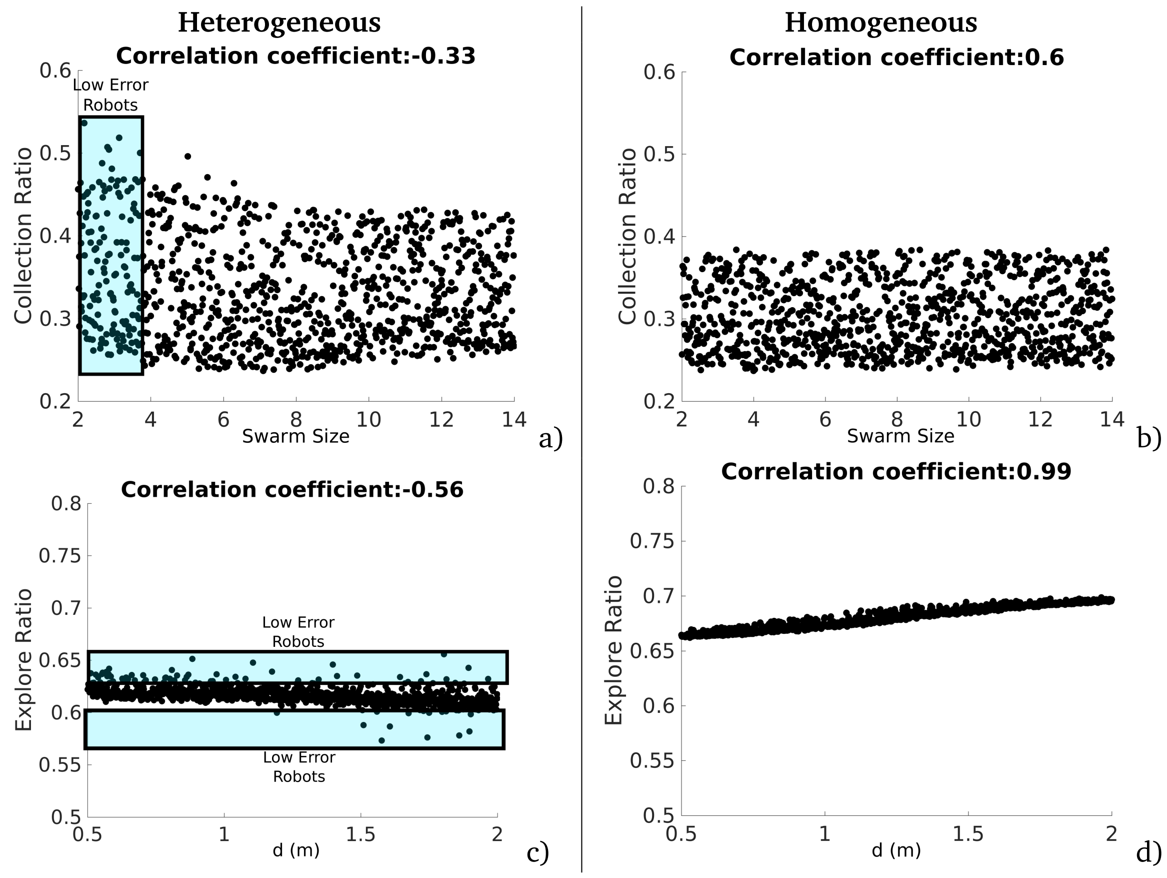


Figure 5.24: LHA for the *d* parameter for CPS with heterogeneous (left column) and homogeneous (right column) errors. Fluctuations introduced by low error robots affect estimation of the correlation coefficient.

Non-dominated Sorting Genetic Algorithm II

Most values for *epsilon* are close to 0 for the heterogeneous error except when *d* is 0.5 and *swarm size* is 14 robots (Figure 5.27a). This means that when the density is high there is no point of using the $\epsilon - greedy$ algorithm because robots are spending too much time avoiding each other and the error accumulates as consequence. The *memory factor* is slightly correlated with *d* where as *d* increases the *memory factor* decreases (Figure 5.27c). In addition, the *memory factor* regulates the *collection ratio* where as the *memory factor* increases the *collection ratio* increases (Figure 5.27c).

Results for homogeneous error are shown in Figures 5.27b, 5.27d, 5.27f and 5.27h and described next. The large amount of noise in the homogeneous error CPS harms optimization for *collection ratio* and *explore ratio*. Therefore, only *total items collected* is optimized by identifying the suitable *swarm size* In other words, the homogeneous error has a huge variation and because of this it is hard for the robots to find a good P that minimizes cost.

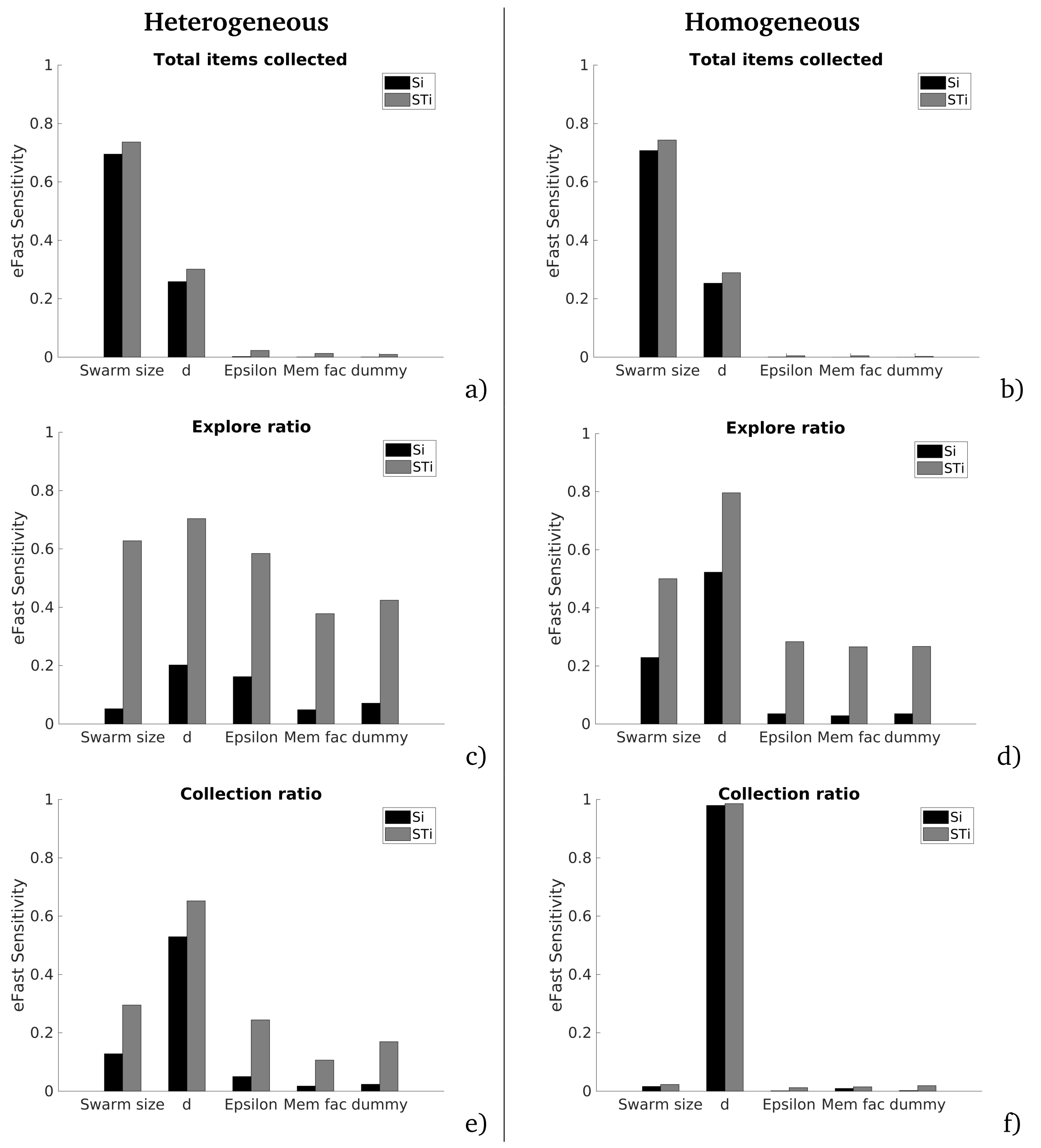


Figure 5.25: eFast results for CPS with heterogeneous (left column) and homogeneous error (right column).

Conclusion

In conclusion, for the homogeneous error CPS appears to have a large impact at the variance for the the different outputs as shown in the results from the eFast. This means that CPS behaves more like NPS and this is more evident in the results from NSGA-II where the best values for ϵ and the memory factor are the ones that have least effect on the optimization (values close to 1). This variance in the homogeneous error affecting robots performance with CPS is explored in more detail in the next section. As for the heterogeneous error, the best value for epsilon is 1 which means that for the robots its better to exploit than to explore.

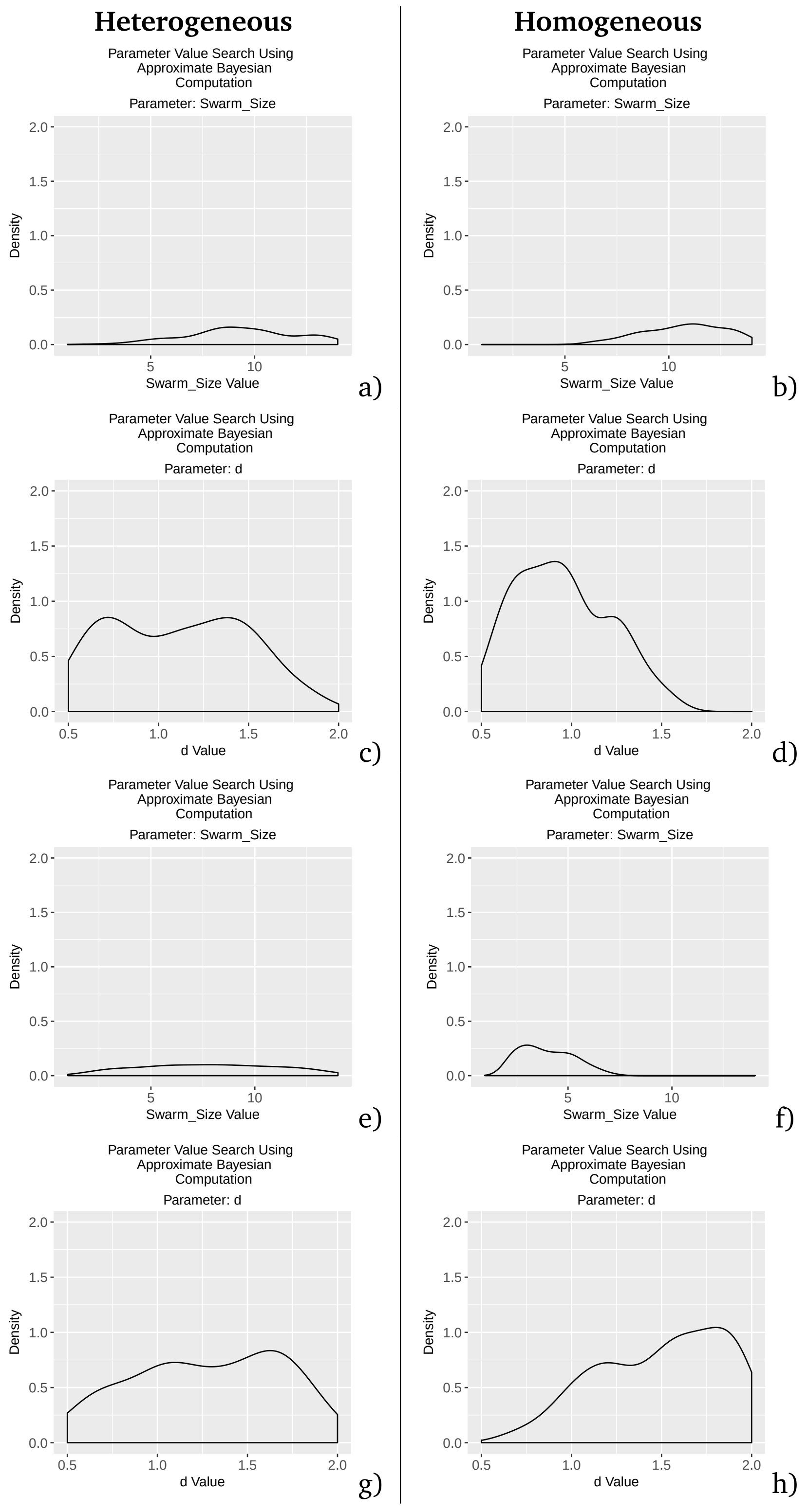


Figure 5.26: easyABC results for CPS with heterogeneous (left column) and homogeneous (right column) error for the *total items collected* output with the *swarm size* (a and b) and *d* parameters (c and d), and for the *explore ratio* (e and f) and *explore ratio* (g and h) for the *d* parameter.

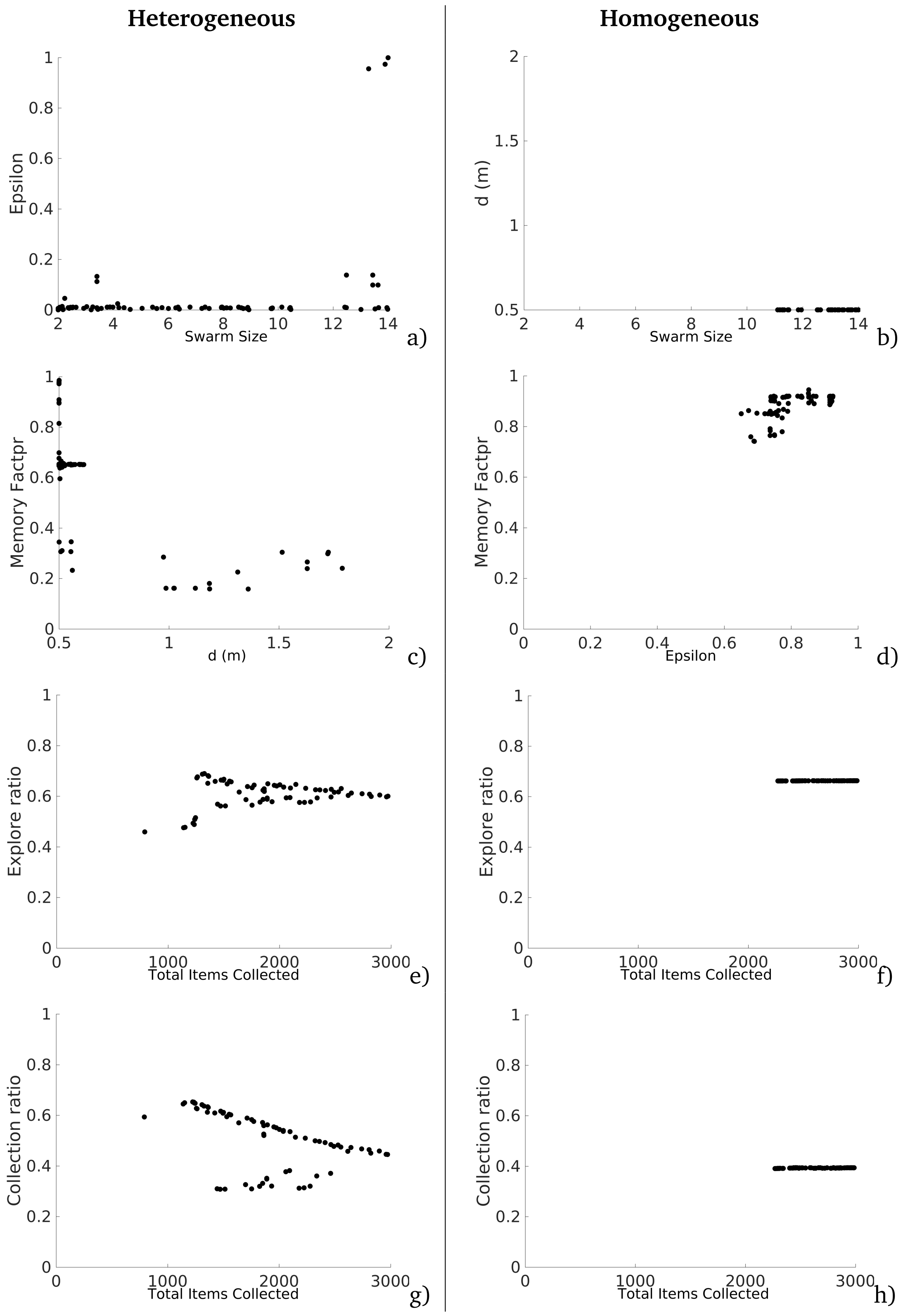


Figure 5.27: NSGA-II results for CPS with heterogeneous (left column) and homogeneous (right column) error. Two dimensional inputs are shown as *swarm size-epsilon* (a and b) and *d-memory factor* (c and d). Two dimensional outputs are shown as *total items collected-explore ratio* (e and f) and *total items collected-collection* ratio (g and h).

This is in not the case when the environment is too crowded the best value for ϵ is 1. For the memory factor, it changes a long d and it regulates the *collection ratio* output.

5.5 Summary

It is common assumption that all the robots are similar and a single robot model in simulations represents the entire group of robots in robotic swarms. Furthermore, it has been thought that all robots would have the same behaviour when performing a task, however, as it was shown in this chapter that it is not always the case.

At the moment of retrieving the trajectories for each robot it was shown in Section 5.2 that each robot experience a different degree of error. Each robot has a different bias, small or large, of moving towards the left or right direction. Then, the trajectories of 14 robots are recorded and modelled.

In the enriched analysis in this chapter it has been shown that two different patterns emerged from heterogeneous and homogeneous errors for each strategy. Each pattern represents different properties of the task partitioning approach. On one hand, the pattern shown with homogeneous error provides information of the interactions of each parameter with each output. On the other hand, the pattern with heterogeneous error provides information how the swarm copes with robots with high and low degree of error for each strategy.

In the next chapter heterogeneous and homogeneous error affects the swarm from a micro perspective is studied. The results include the task specialization of each robot according to its error and how diversity changes with each swarm size. Lastly, results with physical robots are shown.

Chapter 6

Study from micro perspective

6.1 Introduction

In the previous chapter the effect of considering individual robot errors in emulation with each strategy is explored. An emulator was used to aid with this study and it was found that results differ from heterogeneous and homogeneous error. In this section this discrepancy is explored in detail by using experiments from a micro perspective with simulations and hardware (stages 2 and 3 from the experimental framework shown in Figure 5.6).

The second stage consists of studying the effects of heterogeneous and homogeneous error from a local perspective, considering the contribution of each robot by itself to the task in simulations. The third stage consists of experiments with physical robots in order to validate the results from emulation and simulations.

The parameter values chosen, unless stated, for the experiments in this section are shown in Table 6.1. A screen-shot from a simulation experiment is shown in Figure 6.1.

The content of this chapter is the following. Firstly, the impact of the arena size in the navigation of the simulated robots is explored in Section 7.2. Then, the difference in performance between heterogeneous and homogeneous error through the convergence of P is described Section 6.3. Different flexibility properties for DPS and CPS are described in

Table 6.1: Parameter values for the micro perspective experiments

Parameter	Value
Experiments length	5 hours
Swarm size	6
d	1 m
P	0.5 m
α	0.5
ϵ	0
Memory factor	0.25

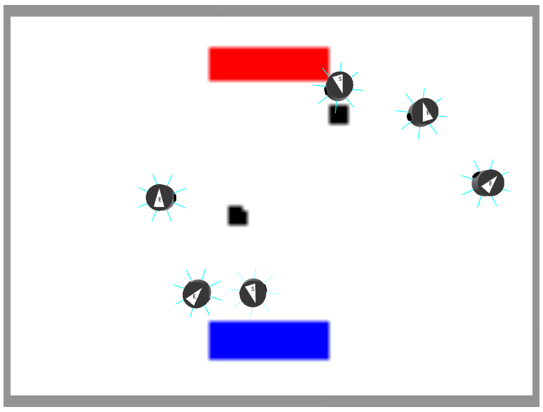


Figure 6.1: Screenshot of one of the experiments in simulation. The red rectangle represents the home area, the blue rectangle represents the items source and the small black circles represent items.

Section 6.4. Section 6.5 analyses the change of diversity in each strategy. Experiments are validated with physical robots in Section 6.6. Lastly, Section 6.7 gives a closing summary of the chapter.

6.2 Impact of arena size in robot navigation

Results from the previous chapter shown that parameter *d* defines the collection ratio for the swarm. This section describes how the width of arena (l) affects the performance of the swarm. Moreover, robustness analysis technique is used here to identify the appropriate values for ϵ and the *memory factor* for CPS, and *swarm size* that exhibit the best item collection. In some scenarios, a robot in the *go to source* state is unable to reach the target due to the robot being trapped or the target being unreachable because of obstacles obstructing the path. In order to prevent the robot being trapped in the same state for a long a time, a timer is introduced. If the distance between the target and the robot does not decrease after the timer is overcome, the robot switches to the *explore* state.

The value assigned to the timer is of great importance. On one hand, a small value causes the robot to switch state for every obstacle found which is undesirable because affects the estimation of P_i . On the other hand, a robot spends a large amount of time trying to reach a unreachable object which leads to an item collection degradation over time.

In an arena of 1.0 x 1.3 m where *d* is 1.0 m a swarm of 6 robots are performing NPS. Two different values, 10 and 30 seconds, are chosen as timers to study how this affects the navigation of the robots.

With a timer of 10 seconds the robots switch state immediately from *go to source* to *explore* state as shown in Figure 6.2a. Therefore, more time is allocated to the *explore* state. In addition, time in the *go to source* state is smaller than *go to nest* which means that the robots are not reaching the estimated target. The reason is that, in theory, time spent in the *go to nest* state should be the same as the *go to source* but in practice it might be slightly different due to any obstacles obstructing the trajectory.

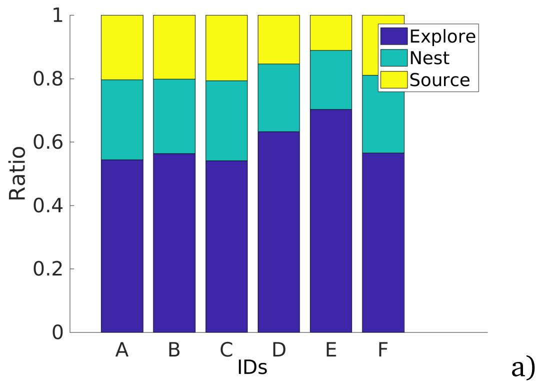


Figure 6.2: Individual performance for different timers, 10 (a) and 30 seconds (b) with robots performing NPS. *Explore ratio* decreases and *go to nest* ratio increases as the timer increases.

As the timer increases and a value of 30 seconds is selected, robots spend more time in the *go to source* state as shown in Figure 6.2b. Time spent in the *go to source* increases because the robots are spend 30 seconds trying to reach a target that is outside of the arena boundaries. This decreases time spent in the explore state.

This timer not only affects the amount of time spent in each state but it also affects the performance in terms of items collection in task partitioning. Figure 6.3a shows that when the timer is small (7 seconds) there is no improvement when the robots performing SPS are travelling short P. Therefore, the best item collection is provided by the robots travelling a d of 1.0 m. In Figure 6.3b the timer value is 60 seconds and the best item collection is provided with a short value of d of 0.3 m. The main reason is that long P values cause robots to spend large amounts of time in the *go to source* state trying to reach a target that is outside of the boundaries.

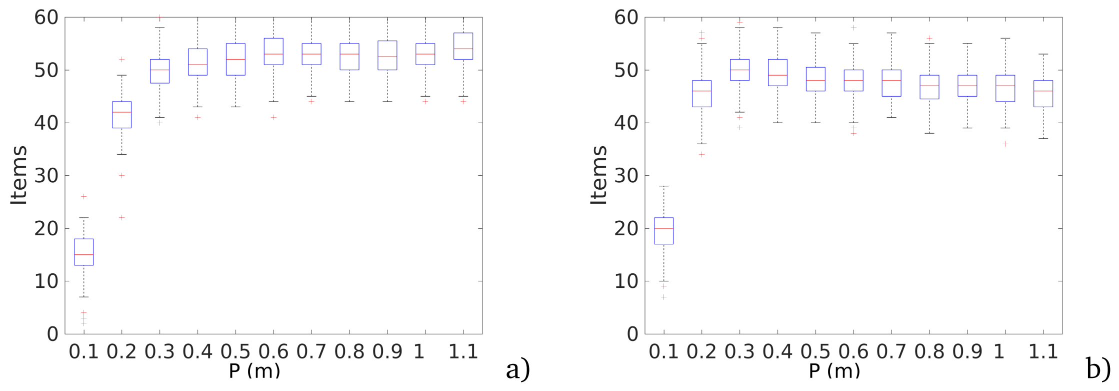


Figure 6.3: Item collection with different lengths of P with robots performing SPS with timers of 7 s (a) and 30 s (b). The best item collection is 1.1 m for a timer of 7 s and 0.3 m for a timer of 30 s.

Robots in an arena where this timer has no impact should spend roughly the same amount of time in both states *go to nest* and *go to source*. In order to identify arena size where the item collection is not affected by this timer, a robustness analysis evaluates different values for the timer with different arena sizes as shown in Figure 6.4. The robustness analysis is used here to assess the local sensitivity and detect in the range a parameter produces significant changes to the output. For the arena size of 1.0 x1.3 m the A-test score shows that the value chosen for the timer has large difference (A-test score greater than 0.73) whenever is compared to the baseline value of 10 seconds. However, as the size increases in terms of width (l) the impact decreases until the difference in the A-test score is small (A-test score less than 0.73). This means that the timer has no effect on the item collection and can be ignored for future experiments.

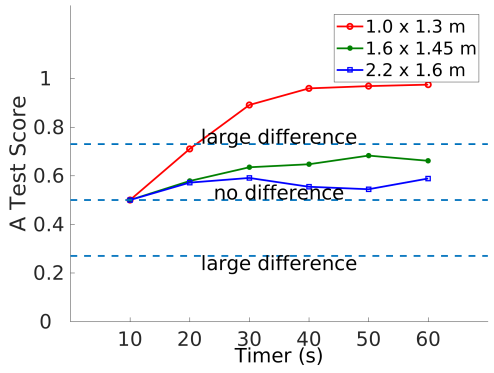


Figure 6.4: Robustness analysis for the timer parameter with different arena sizes. As the arena size increases, the item collection is less affected by the timer.

Lastly, in order to identify the appropriate swarm size for the rest of the experiments in this section a robustness analysis again was used with different swarm sizes performing DPS as shown in Figure 6.5. Only when there is a single robot collecting items the item collection does not improve as α changes. Therefore, a swarm of 6 robot is more than enough to visualize the changes of item collection with different values of α.

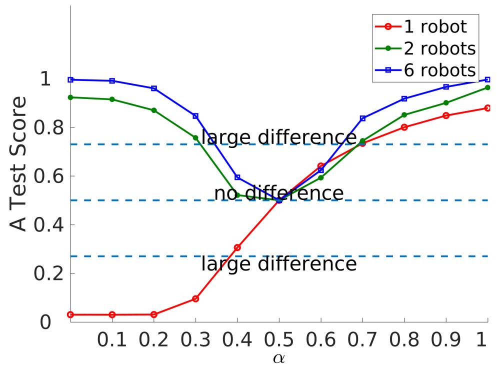


Figure 6.5: Robustness analysis for the α parameter with different arena sizes. Only when there is a single robot α contributes negatively to the item collection.

A robustness analysis is performed to ϵ and memory factor parameters for CPS in order to measure their degree of sensitiveness of the item collection at the moment of changing them. There is not change for the *memory factor* parameter, however, ϵ is more sensitive. The value that provided the best items collection is 0. The values of 0 for ϵ and 0.25 for *memory factor* are used for the experiments in Chapters 5 and 6.

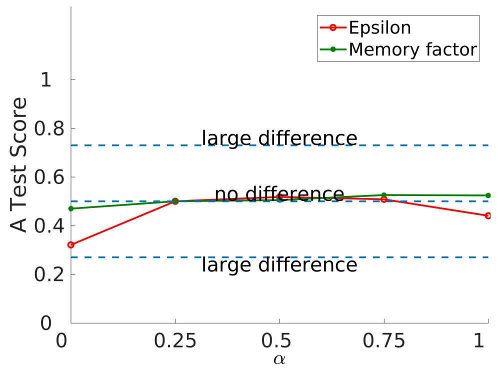


Figure 6.6: Robustness analysis for epsilon and memory factor parameter for CPS.

As conclusion, it is important to identify the appropriate arena size that allows robots to perform their navigation without being disturbed by the arena boundaries. In addition, the best values for *swarm size*, ϵ and *memory* are identified for the experiments in this chapter.

6.3 Heterogeneous and homogeneous errors

As discussed in Section 5.2 when modelling the error for each robot, it was found that the error varies between robots. This section describes error diversity and uniformity in simulations.

In the first set of experiments, the robots are performing DPS where P_i is the same as *d*. Convergence of P_i is shown in Figure 6.7. All the robots converge to a single similar P with homogeneous error (Figure 6.7a and 6.7c). However, robots with heterogeneous error converge to different P_i for each robot i (Figure 6.7b and 6.7d). In addition, robots performing direct transference (Figure 6.7d) take 1 hour longer to converge than with indirect transference (Figure 6.7b).

In the previous section it was noted that results between NPS and CPS with homogeneous error are similar, as solutions with homogeneous error have a high degree of fluctuation. Figure 6.8a illustrates the distributions of P_i for robots B and E after 6 hours. Since robot E has a greater error it selects P_i smaller than 0.75 m, because the probability of finding items increases. Time spent transporting items increases. Robot B can travel a longer P_i because its smaller drift. However, for the homogeneous error (Figure 6.8b) the distribution is very similar between robots and it is uniform. This means that it is hard or nearly impossible to find the appropriate P_i to travel, as sometimes drift can be small and other times big.

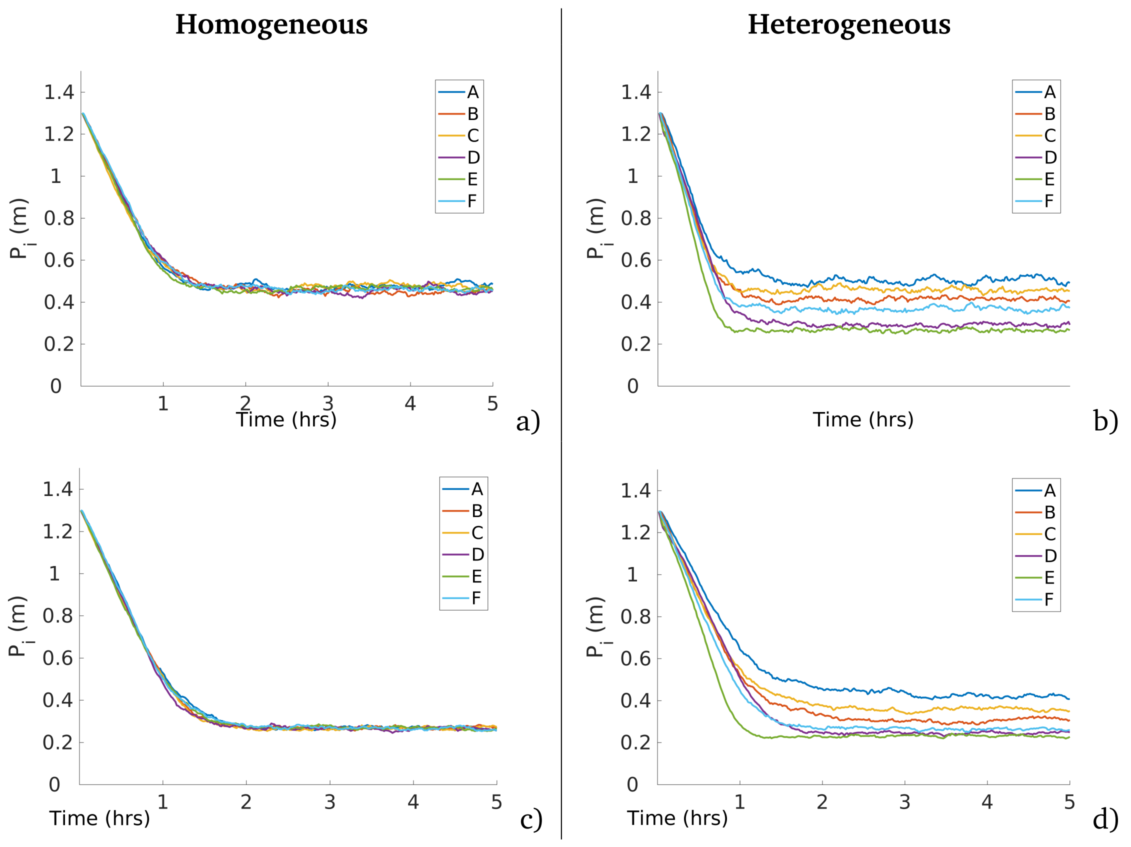


Figure 6.7: Convergence to solution with homogeneous (left column) and heterogeneous (right column) errors with DPS with direct (first row) and indirect (second row) transference. Robots with homogeneous error converge to a single individual partition length (P_i) and robots with heterogeneous error converge to different P_i .

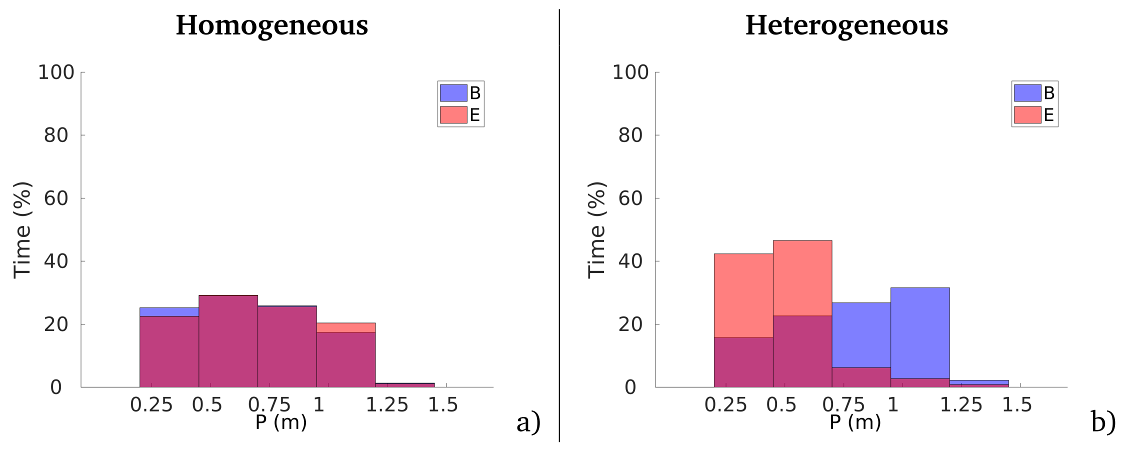


Figure 6.8: Histogram of solutions chosen by each robot with homogeneous (a) and heterogeneous (b) errors with CPS.

The impact of different errors is reflected also in the individual performance. In the experiments shown in Figure 6.9 all the robots start with a P_i of 0.4 m and α equals to 0.5 with homogeneous and heterogeneous error models DPS. Figure 6.9a illustrates the robots with homogeneous error, where each robot has a similar individual performance with each other.

On the other hand, robots with heterogeneous error have different individual performance according to each robot as shown in Figure 6.9b.

Robot specialization emerges from DPS in sense that every robot learns its own P_i according to their inherent degree of error. This is consistent with the results presented in Figure 6.9 where P_i for each robot is correlated with the amount of time that robot spends exploring the environment. For example, robot A spends the least amount of time in the explore state and, hereby, has the greater P_i with a value close to 0.5 m. Robots D and E spend the greatest amount of time in the explore state and these robots travel the small P_i with a value close to 0.3 m. In addition, this is consistent with the trajectories shown in Figure 5.3.

For instance, Figure 6.8 shows that robots D, E and F learn the smallest P_i , and this related with their individual performance where these robots spend more time in the *explore* state than the rest of the robots as shown in Figure 6.9. In addition, Figure 5.2 shows that these same robots have bigger error drift.

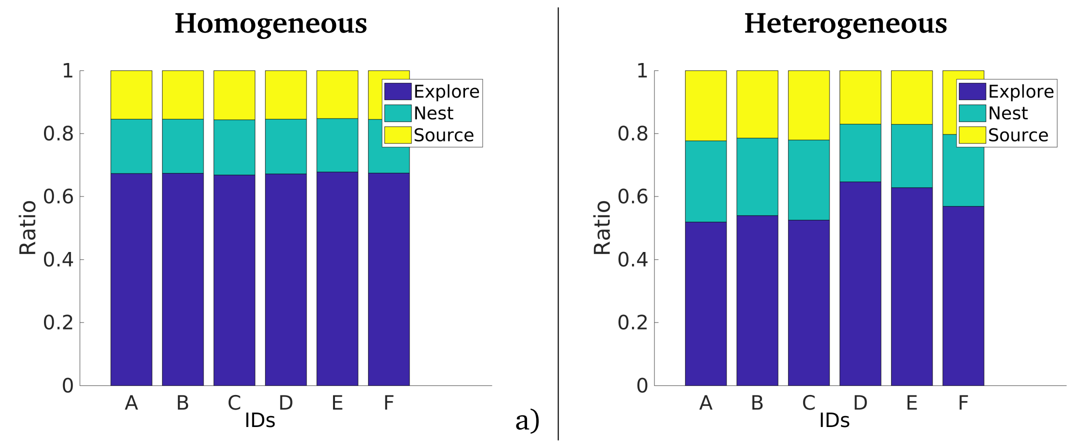


Figure 6.9: Individual performance with homogeneous (a) and heterogeneous (b) error models. Robots with homogeneous have a similar individual performance and robots with heterogeneous error have different individual performance.

It is important to bear on mind the difference in performance between heterogeneous and homogeneous errors, in order to select the appropriate value of α for the swarm to maximize the item collection as shown in Figure 6.10. In case of homogeneous error model α should be 0.4 in order to have the best item collection. As with heterogeneous error model that value would be 0.5.

Indirect item transference is the method chosen for most of the experiments shown in this chapter because it has the highest item collection with the lowest convergence speed. This is an important aspect to consider because the battery of physical robot only last 30 minutes.

In conclusion, it is vital to consider individual models for each robot when working with swarm robotics. In this way it is possible to make the reality gap smaller and be able to choose the best α that maximizes the item collection and, in addition, potentially select the best P_i and α for each robot to improve the item collection.

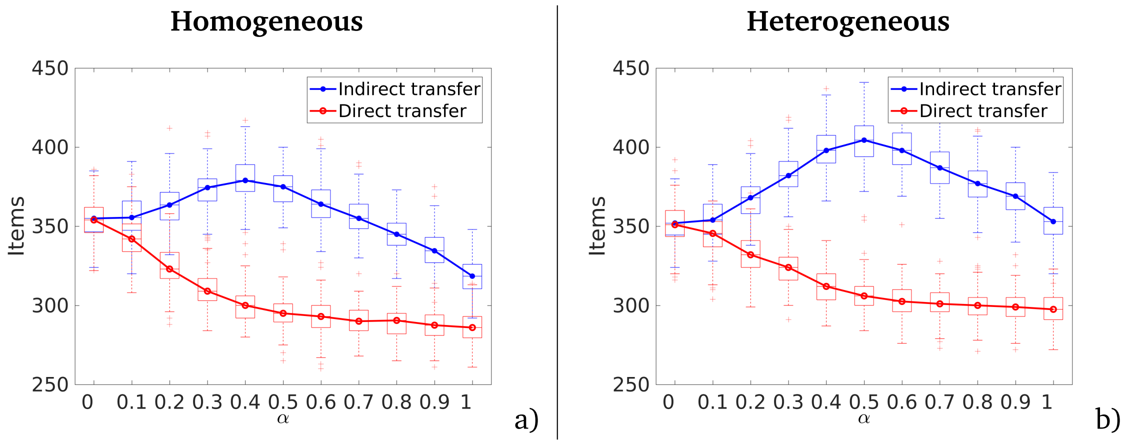


Figure 6.10: Item collection with homogeneous (a) and heterogeneous (b) errors with different values for α. The peak for item collection is 0.4 for homogeneous and 0.5 heterogeneous error model.

6.4 Flexibility

In this section, flexibility is studied by analysing how the robots adapt P_i with DPS and CPS when the parameters for individual error are swapped, the swarm size reduces or d changes decrease during the simulation. The objective is to provide a further understanding in terms of flexibility for each strategy.

The value for α selected is 0.4 and this only because with this value it is easier visualize the flexibility properties.

6.4.1 Inherent parameters error swap

A characteristic that DPS and CPS share is the ability of finding and adapting P_i to the given environment, inherent error and swarm size. In these experiments, the robots adapt their P_i when the individual error changes for each robot. In order to achieve this, the parameters for the error model for each robot are swapped with its counter part after 2.5 hours simulation time threshold is reached. This means that the parameters of the robot with lowest degree of error, in this case robot A, are changed for the parameters of the robot with highest degree of error, robot E, and vice-versa. This is done for each robot across the swarm and the results are the following.

Figure 6.11 shows how the robots adapt when the error parameters are interchanged with each other for indirect and direct transfer with DPS and CPS. The robots with DPS are able to adapt to this sudden change successfully and just after 3.5 hours the robots reached a steady state for the new error.

The results for CPS differ between indirect (Figure 6.11a) and direct transfer (Figure 6.11b). Results with indirect transfer (Figure 6.11a) does not seem to change and this might because of the learning factor which causes the swarm take longer to adapt as this was explained in the previous chapter.

For the direct transfer (Figure 6.11d) P_i increases steadily for all the robots. Since the robots have to wait for each other to exchange items, it is more time efficiency to travel a longer distance than wait multiple times. This consistent with results reported in Section 4.3.3.

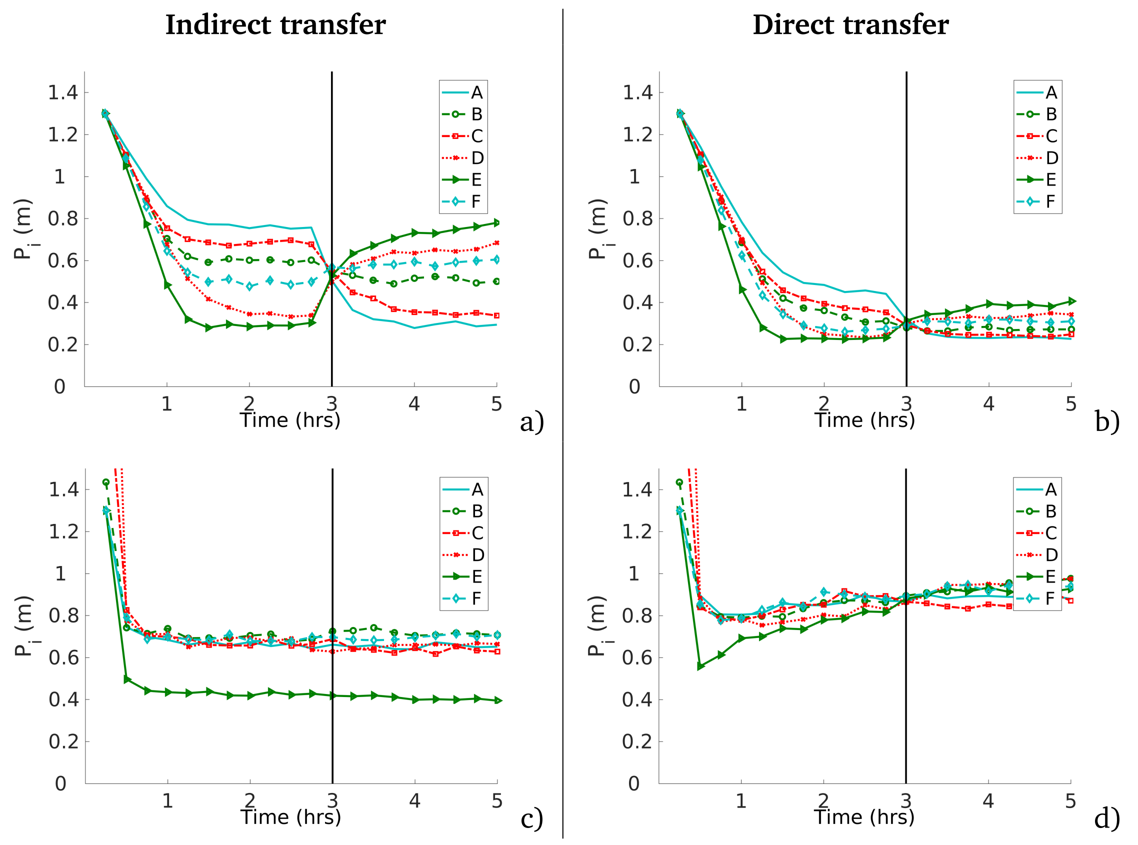


Figure 6.11: Inherent error parameters are swapped between robots convergence graphs with indirect and direct transfer (left and right column) for DPS (first row) and CPS (second row). The parameters for the error model are interchanged after 2.5 hrs time point is reached.

Task specialization emerging from the inherent characteristics from each member can be found in *Pheidale* ants, where ants task specialize according to their size. Large ants carry items to the nest whereas small ants do brood care. (Duarte et al., 2011)

6.4.2 Swarm size reduction

In this section the experiments show how the swarm adapts to a reduction in the number of active robots over time. Robots stop being active when a partial failure is injected in this case, motors stop rotating. Faults are injected after 3, 5 and 7 hours time thresholds are reached and robots F, E and D stop moving in each time point respectively.

The convergence to solution graph for DPS and CPS is shown in Figure 6.12. Robots using indirect transfer (Figure 6.12a) are more resilient to swarm reduction than direct transfer

(Figure 6.12b). P_i decreases slightly for indirect transfer whereas in direct transfer it decreases more with each non-active robot introduced. This is because robots with direct transfer, item transaction is tightly coupled with robot interaction, or in other words two robots have to meet in order to exchange items. Therefore, as the swarm size decreases, the probability of two robots meeting to exchange items decreases. As consequence, this leads robots waiting for long time for a second robot to receive the item and eventually the first robot switches to the *go to nest* state. This increases the distance travelled by the first robot and with this the probability of not finding items.

As for CPS with indirect transfer (Figure 6.12a) it behaves slightly like DPS with indirect transfer. P_i decreases slightly and this can be because there are less robots in the environment. Robots using direct transfer (Figure 6.12a) behave in a similar way than previous study where it is better for the robots to take the items directly to the home area than waiting.

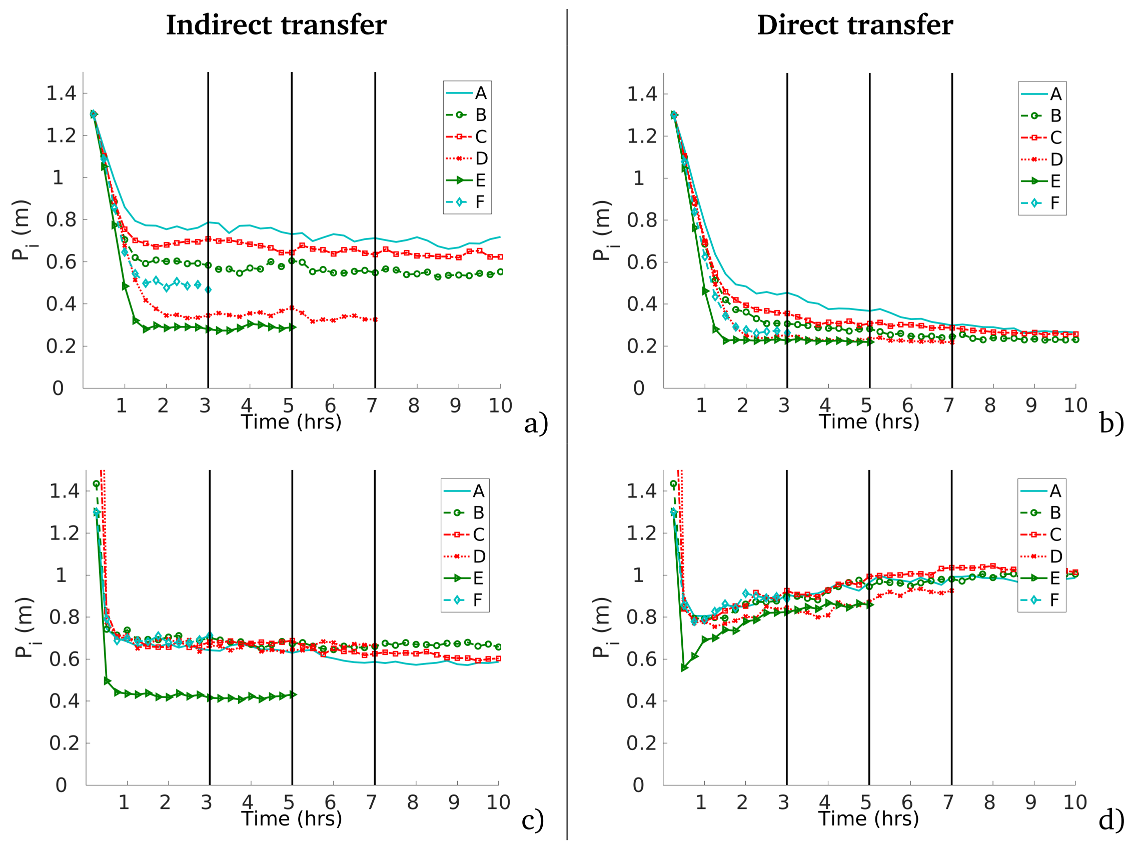


Figure 6.12: Swarm size reduction convergence graph for indirect and direct transfer (left and right column) for DPS and CPS (first and second row). At 3, 5 and 7 hours time points robots F, E and D stop moving.

6.4.3 Distance between home area and items source reduction

The experiments in this part show how the swarm adapts to environmental changes in this case with a changing d. The initial d is of 1 m but every 2.5 hrs. this distance is decreased by

0.2 m. It is important to mention that only *d* changes and the arena size remains the same.

Figure 6.13 illustrates the convergence to solution for DPS and CPS with indirect and direct transfer. In this case both methods of transference behave roughly similar for DPS. As *d* decreases P_i increases for most robots except D and E. This is due to the fact that the degree of error for robots D and E is very high and the probability of finding items is low even for a distance of 0.4 m. Whereas robots A, B, C and F have a higher probability of finding items and P_i increases indefinitely because of this.

Further work could set limits to P in order to prevent increasing exponentially because this could affect the resilience of the robots. For instance, if robot A experiences an increase of degree of error then, it would take significantly more time to adapt to this sudden change. A top limit could be the maximum distance travelled in this case *d*.

In case of CPS P_i for each robot decreases slightly for indirect transfer (Figure 6.13a). As for the direct transfer, P_i increases steadily as shown in Figure 6.14b. This might be because with indirect transfer there are no waiting times like with direct transfer and travelling shorter distance might optimize time spent transporting items by robots. In addition, robots travelling longer P_i is more efficient for direct transfer because time spent waiting is less.

P_i for each robot, except robot E, seem to converge to a similar solution and this might be because it is getting harder for the robots to converge to a good solution and the solutions space is distributed in something similar as a normal distribution (Figure 6.14a). This is because as *d* decreases the probability of finding items increases, therefore robots spend more time travelling and back and forth between the home area and the items source. However, this eventually leads to a bottleneck because many robots are transporting items within a short distance and with this robots spend more time avoiding each other which affects the cost estimation and the error accumulation. This cannot be seen with direct transfer because with this method both robots are required to meet, as consequence there is a better control on the flow of robots moving between the home area and the items source.

6.5 Social entropy

Results from NSGA-II in emulation for NPS showed a discontinuous pareto front for the heterogeneous error and a continuous pareto front for the homogeneous error (Figure 5.12a and 5.12b). In this section, we explore in closer detail the reason of this discrepancy by using the social entropy metric. This metric measures the error diversity across the swarm and described in Section 3.7.2.

Decision trees and nearest neighbour classifiers are used to categorize the robots into two groups according to their individual performance. Principal Component Analysis (Wold et al., 1987) is used to convert the variables from the individual performance data to their principal

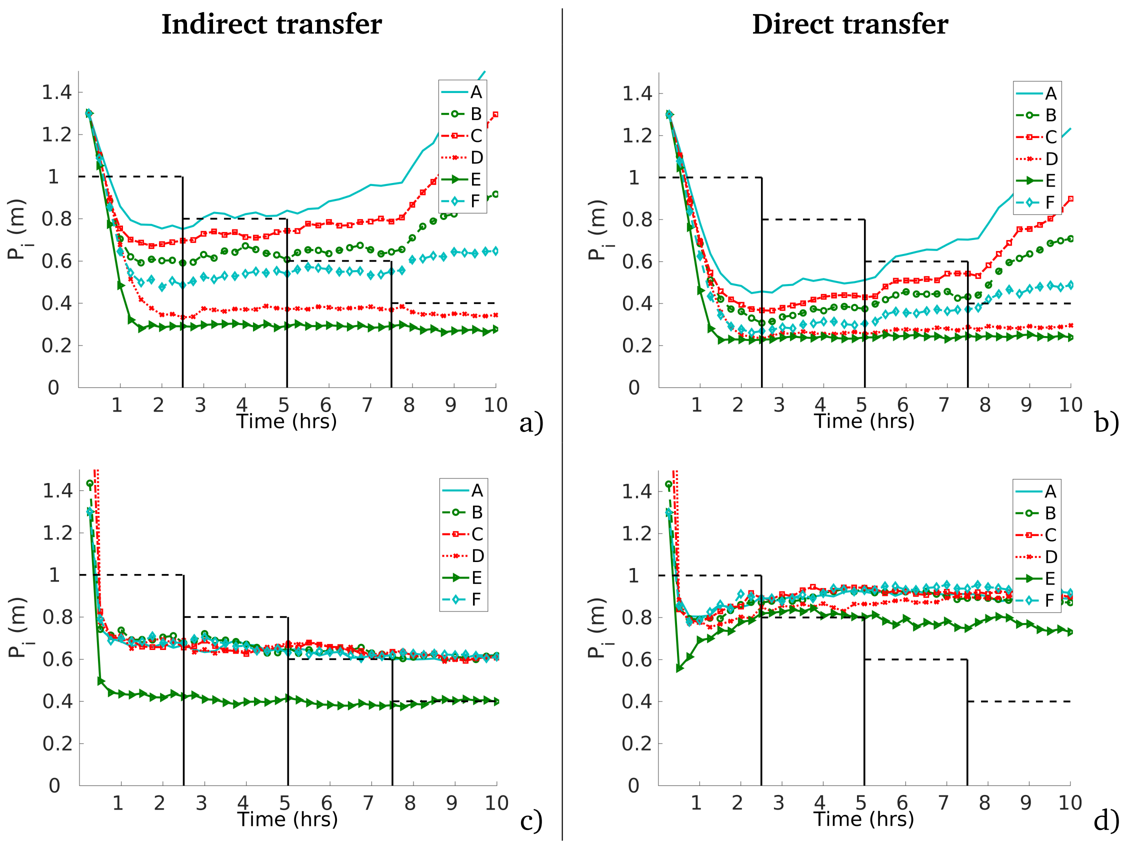


Figure 6.13: Distance home area and items source reduction convergence graph for indirect and direct transfer (left and right column) for DPS and CPS (first and second row). At 3, 5 and 7 hours time points *d* decreases 0.2 m therefore distance changes to 0.8, 0.6 and 0.4 m in each time point respectively.

components. This is done in order to transform possibly correlated variables into a set of values of linearly uncorrelated variables. The requirement for classification is that the accuracy should be higher than 90%. Figure 6.14 illustrates the social entropy for each swarm size in the interval [2,14] for each strategy. The social entropy is overlapped with the results from the NSGA-II reported in Section 5.4.1.

The social entropy for NPS, shown in Figure 6.14a, illustrates that the pareto front breaks when the social entropy highest peak is reached. For the swarm interval between 2 and 4 robots the social entropy is low. This means that the swarm is homogeneous composed with robots with low degree of error. This is the reason for the cluster of points in *d* of 1.75 m. However, after the first robot with high degree of error is introduced (robot D or fourth robot), the social entropy increases. Still, the swarm is homogeneous enough to have the pareto front continuous. However, the discontinuity in the pareto front appears when the second robot with high degree of error is introduced (robot E or fifth robot). The social entropy reaches its highest peak point at this point. The social entropy starts to decrease after the sixth robot (robot F) is introduced. The social entropy decreases steadily because more robots with low degree of error are added to the swarm and this decreases the diversity. The pareto front resumes after the ninth robot is introduced. After this point the pareto front is similar to the homogeneous error pareto front (Figure 5.12b).

Overall, the social entropy across the swarm for SPS is similar than with NPS as shown in Figure 6.14b. The social entropy peak again is located after the fifth robot and the pareto front becomes discontinuous after this point. However, in contrast to NPS the social entropy remain high (greater than 0.8). This prevents the pareto front being continuous until the social entropy drops below 0.8. SPS creates a clear distinction between high and low degree of error and because of this, the social entropy increases.

Results from DPS show that social entropy is even greater than with SPS. This means that DPS divides the swarm size into two groups of robots. This might be because task partitioning is sensible to high noisy degree of error robots.

Finally for CPS, the social entropy is exactly similar as than with SPS. However, a characteristic that CPS has is that there are no discontinuities in the pareto front. This lead us to speculate that this might be related with the learning mechanism. In CPS, each robot tries to find the P_i that provides it the best performance. Therefore, optimality can be achieved for any swarm size. Whereas, DPS and CPS does not guarantee that the selected P_i provides the best performance for each robot.

As just mentioned the social entropy changes with each strategy and the reason is that each strategy provides different performance for each robot. For NPS, the highest degree of error robots classified are 2, robots D and G. SPS and CPS decrease the error in the robots, and with this, the amount of time spent in the explore state, however the number of robot classified as high degree error increases (D, E and G). DPS, for this scenario, the robots experience the least amount of time in the explore state. The number of high degree error robots increases to 4 (D, E, G and J).

In case of DPS, robots D, E, G and J experience the highest degree of error. Therefore, this robots can not be any be further optimized. This is consistent with the results shown in Figure 6.11 where the P_i for robots D and E with a *d* of 0.4 m does not increase indefinitely like robot A, B, C and F. This means that robots have not reached the optimal distance that decreases the error. In other words, robots D, E, G, and J experience such a high degree of error that the partitioning strategies are unable to provide a suitable P_i for the robots to increase their performance.

6.6 Experiments with physical robots

In this section the experiments studied in the previous section are explored with physical robots. Three different arena sizes are used for the experiments with 6 robots where individual performance and convergence are explored. Sample screenshots experiments are shown in

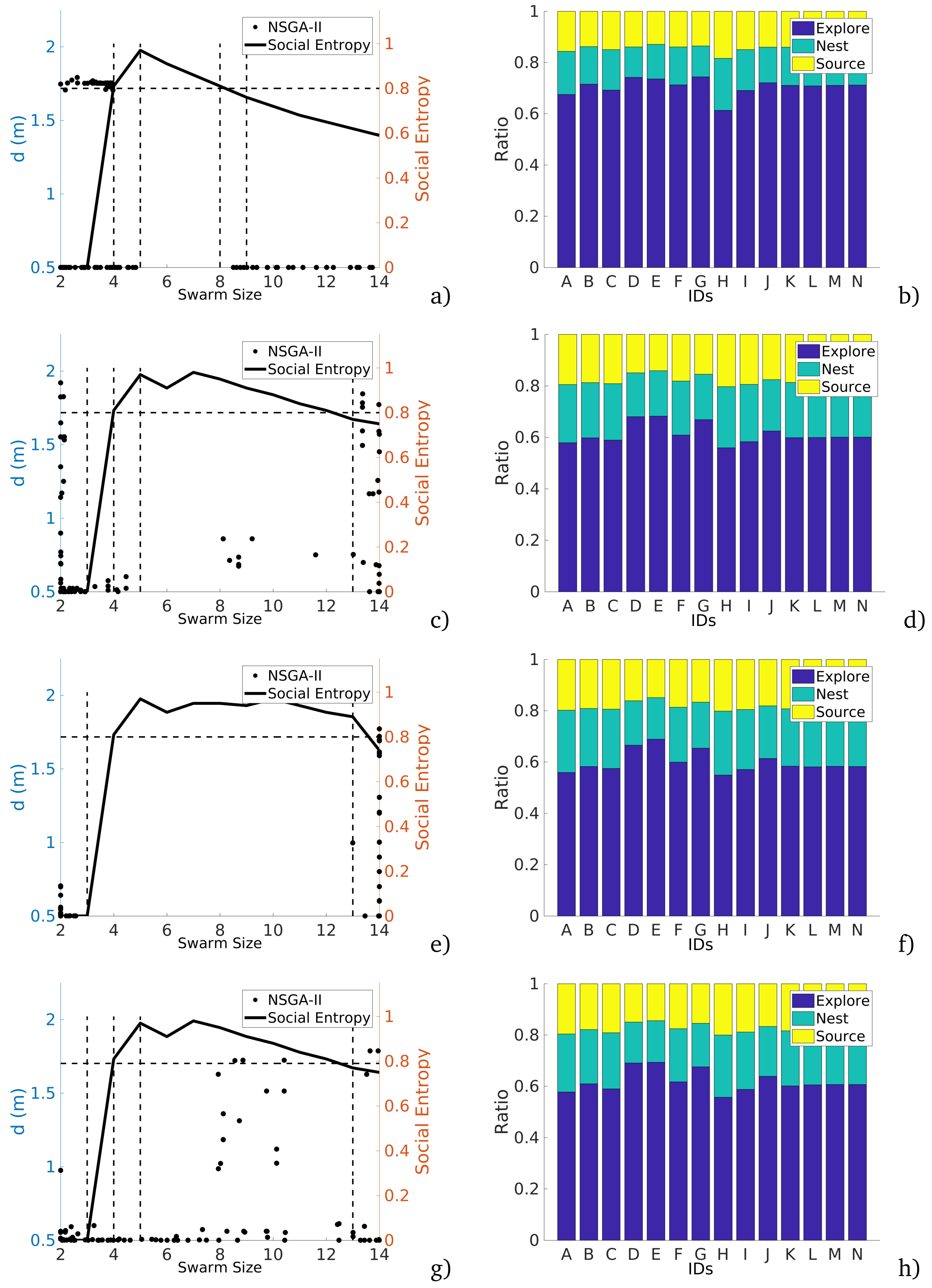


Figure 6.14: Social entropy and individual performance (left and right column) for each strategy: NPS, SPS, DPS and CPS (first, second, third and fourth rows respectively). Social entropy changes for each strategy due to the number of robots classified as high error.

Figure 6.15 and Figure 6.16 represents an item holding an item and a robot carrying an item.

Results for robots performing NPS are shown in Figure 6.17. Regardless of the arena size the robots experience different individual performance which means that they are not identical to each other. In addition, robots spend most of the time exploring instead of retrieving items to the home area. Finally, robots spend less time in the *go to source* than *go to nest* state as shown in Figure 6.17a. This is because since robots are travelling a longer P_i they experience a greater drift from the original position causing the target to be beyond the walls of the small arena as seen before. This effect decreases as the arena increases because the target is within the arena boundaries and shown in Figure 6.17c and 6.17e.

Individual performance for robots with DPS can be found in Figure 6.18. In a similar way than experiments with NPS, robots performing DPS experience an increment in the amount of time spent in the *explore* state as the arena size increases. In contrast with NPS, robots spend more time in the *going to source* state. The reason is that the robots are travelling shorter P_i which decreases the probability of the target being beyond the arena size for the small arena in a similar way that in the simulations.

Since the current battery of the robots last 30 minutes, the convergence experiments were divided into three different sets shown in Figure 6.19. Each set the robots started with a different P of 1, 0.7 and 0.4 m for an arena of size 2.1 x 1.2 m. All the robots converge to a distance close to 0.4 m that changes from robot to robot. The velocity of convergence is not only related to the degree of error in the robot but also, to the speed of the robot which changes slightly for each robot. Furthermore, from the figures can be seen that there is no change in the P_i at least for the first 5 minutes of the experiments. This is the time that it takes to find the first item.

As for the item collection, DPS performs better than NPS as the arena size increases as shown in Figure 6.20 because of the following. Firstly, as the area of the arena decreases the probability of finding the items source increases even though the collection rate is low which means that partitioning is not required. In addition, robots are spending more time dropping and picking up items with DPS which causes a delay in the item to get to the nest. Lastly, robots spend more time avoiding each other when they are in the *go to nest*.

In conclusion, results from experiments with physical robots are similar to results from simulation. The shape of the arena affects the performance of the *go to source* state mainly for NPS. DPS performs better than NPS as the size of the arena start to increase. Finally, final P_i changes according to the amount of error in the robot i. However, there are some differences due to the reality gap because the models used in the simulator do not provide enough information about the real world.

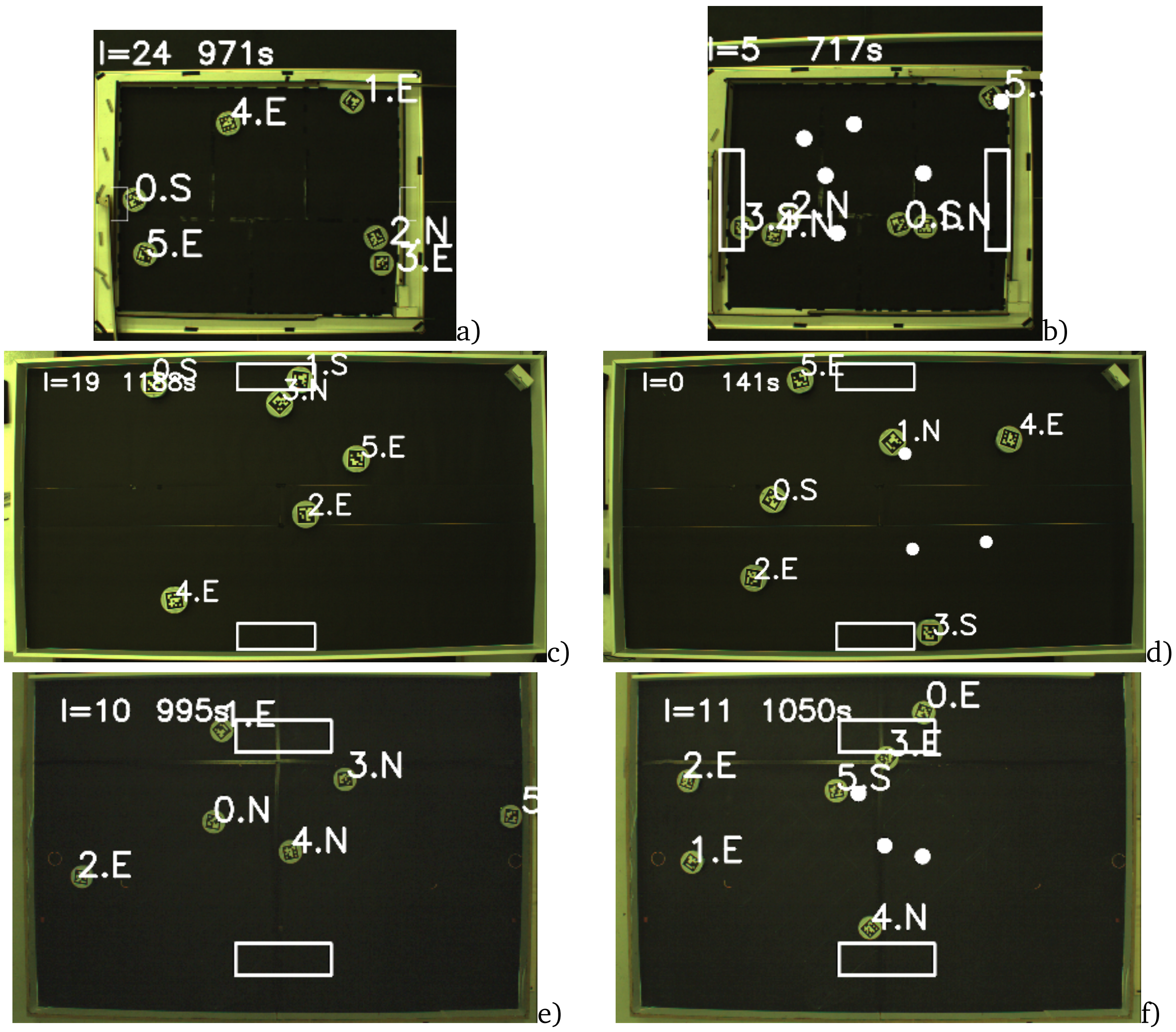


Figure 6.15: Sample screenshots experiments in hardware for NPS (left column) and DPS (right column) for different arenas: 1.0 x 1.3 m (first row), 2.1 x 1.2 m (second row) and 2.4 x 1.6 m (third row). The number next to the robot present the ID of the robot. The character represents the current state: explore (E), go to nest (N), go to source (S) and waiting for transfer (W). The layout of the arena is that the rectangle on top illustrates the home area and the rectangle on bottom represent the items source. White circle represent items.

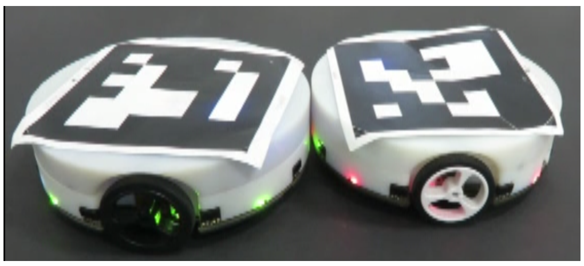


Figure 6.16: Robots exchanging items. The robot on the right with red lights is holding an item and the robot on the left with green lights is not carrying items.

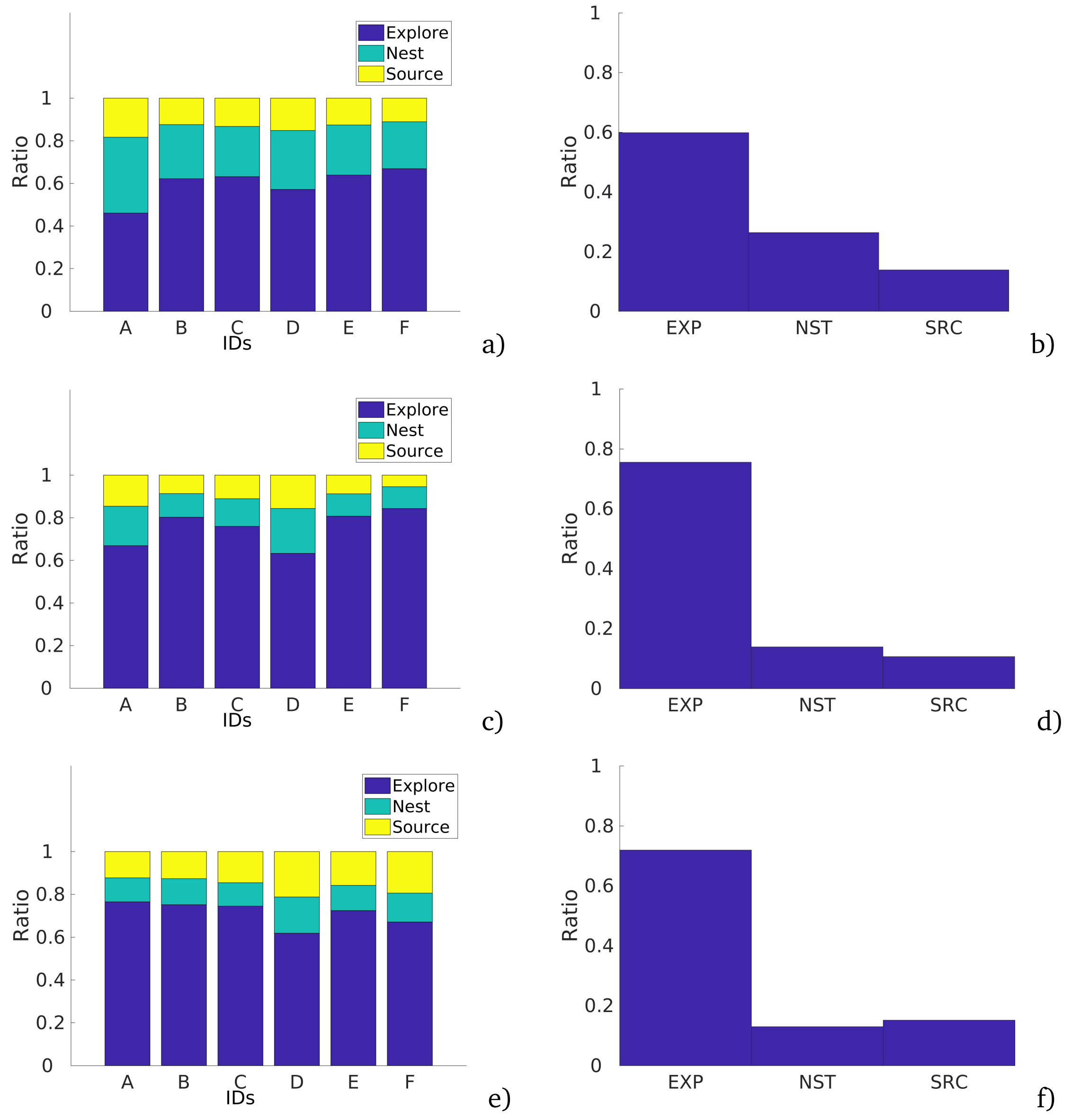


Figure 6.17: Individual performance for physical robots performing NPS for three different arenas with dimensions of 1.0 x 1.3 m (a), 2.1 x 1.2 m (c) and 2.4 x 1.6 m (e). The go to source rate is smaller than the go to nest as the arena becomes smaller.

6.7 Summary

The individual contribution of each robot changes according to its inherent error as shown in Section 6.4. If the homogeneous error is considered, all the robots converge to the same P_i. However, if the heterogeneous error is considered, then each robot converges to a different P_i. P_i is able to adapt to changes in the error itself, d and swarm size.

The social entropy measurement for each strategy bridges the results between the macro and micro perspective (Section 6.5). The number of high degree error robots plays an important role at the moment of defining optimality. As the number of high degree of error

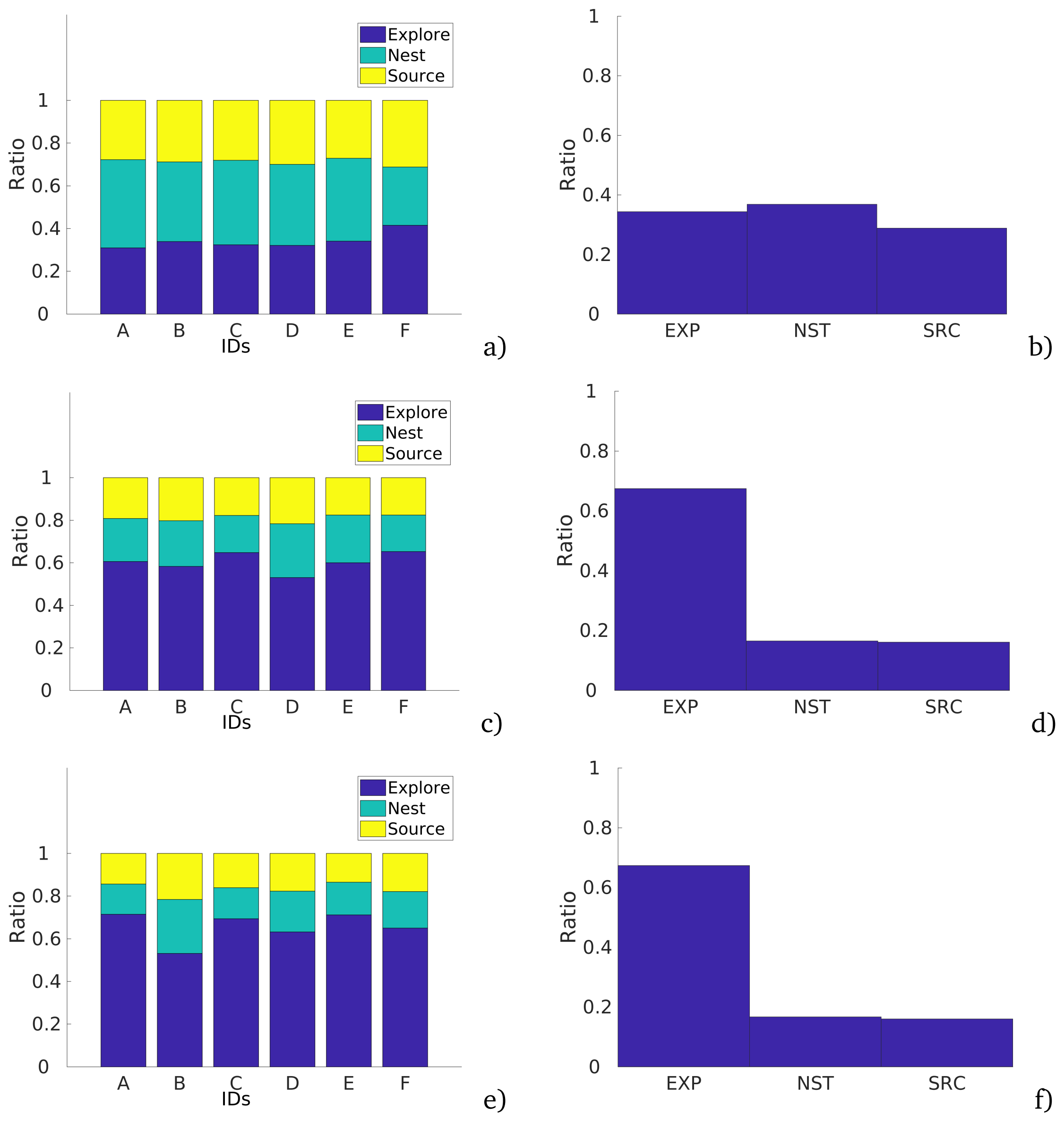


Figure 6.18: Individual performance for physical robots performing DPS for three different arenas with dimensions of 1.0 x 1.3 m (a), 2.1 x 1.2 m (c) and 2.4 x 1.6 m (e). Time spent in the *go to source* state is roughly similar to the *go to nest* state and time spent in the *explore* state is lower than with DPS.

robots increases this affects the range of optimal solution. In addition, even though the task partitioning strategies increase performance for each robot, there are a specific number of robots that gain no benefits of the task partitioning and this is due to their high degree of error.

Finally, the experiments with physical robots shown in Section 6.6 validate the previous results in terms that each robot experiences a different performance and each robot converges to a different solution according to the inherent error.

In conclusion, it is important to consider both, homogeneous and heterogeneous errors, in

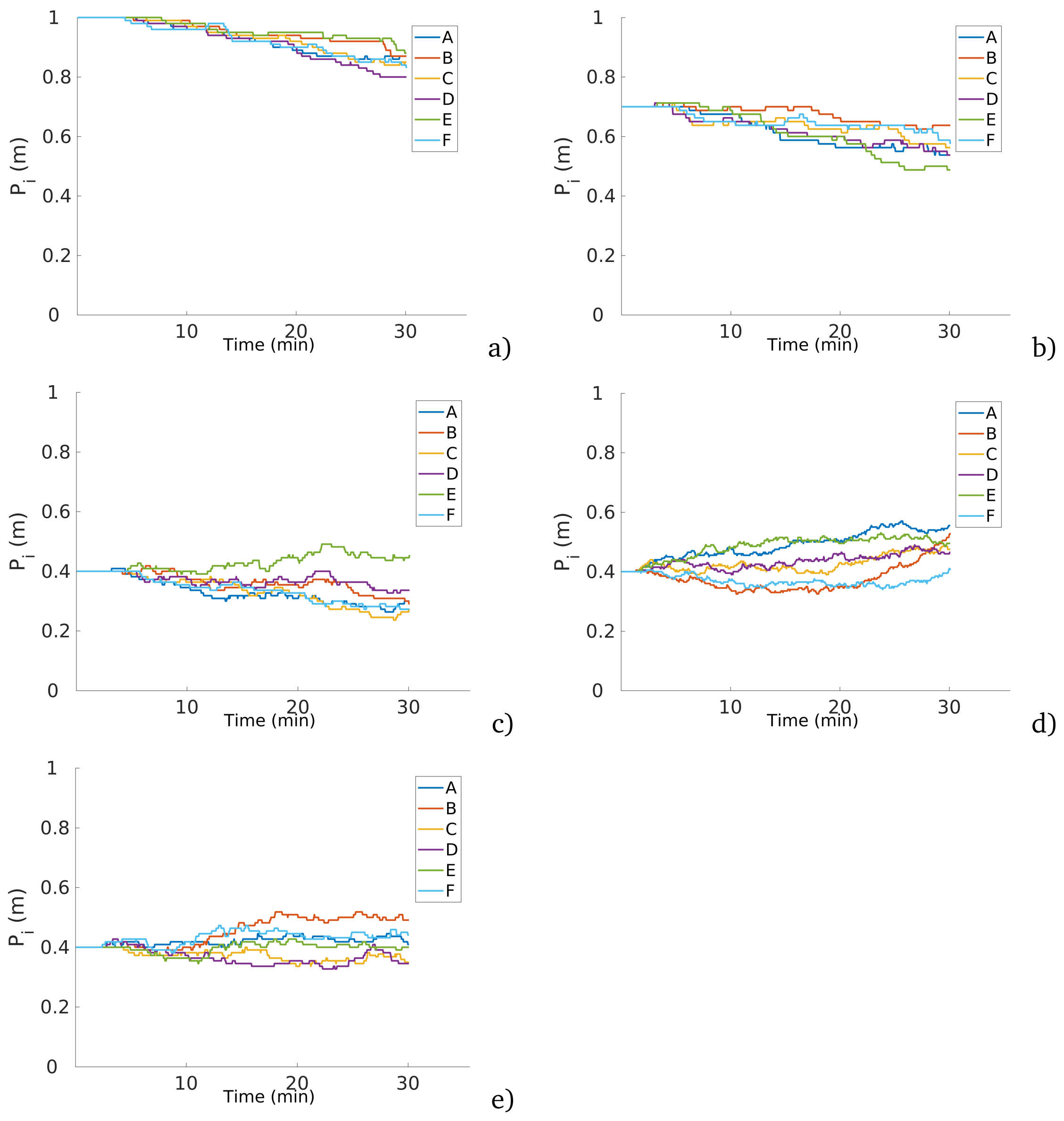


Figure 6.19: Convergence of P_i for different P_i^0 of 1.0 (a), 0.7 (b) and 0.4 m (c) for an arena of 2.1 x 1.1 m. Convergence for arenas 1.0 x 1.3 m (d) and 2.4 x 1.6 m (e) for P_i^0 of 0.4 are also shown. The convergence changes from robot to robot and is due to the error and velocity of the robot.

order to have a comprehensive understanding of the swarm.

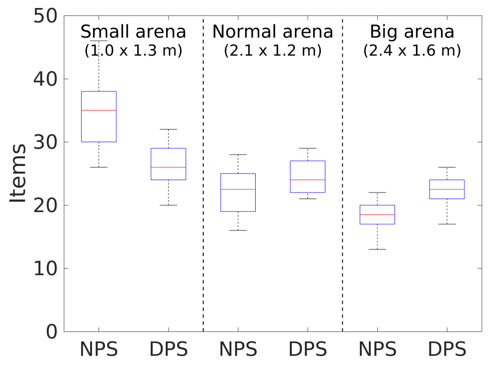


Figure 6.20: Item collection with physical robots and different arena sizes. DPS starts to overcome NPS as the arena size increases.

Chapter 7

Conclusions and Further Work

7.1 Conclusions

Dead-reckoning error is a common problem in robotics generated from multiple factors. It can be classified into systematic and non-systematic errors. Systematic errors are caused by problems or changes in the design of the robot, for example, the difference between wheels size or the weight is not distributed uniformly. Non-systematic errors are introduced by the interaction of the robot with the environment, for example, wheel slippage and small rocks in the gears. Systematic errors, if detected, can be easily compensated, whereas non-systematic errors are hard to compensate due to their non-linear nature.

Dead-reckoning error causes navigation in foraging robots to be inaccurate. In other words, landmarks recorded by a robot would change from the position over time drifting away from the correct position of the target. Hereby, this affects the performance of the robot because the robot reaches a wrong target position when attempting to return to the target.

Task partitioning has been used to reduce dead reckoning error. This is achieved by dividing the main task into multiple smaller components. Instead of a robot transporting an item from the items source directly to the home area, in task partitioning the robot stops at a specific distance *P* when travelling towards the home area. Since the the robot is travelling a shorter distance *P* the error accumulation is smaller and navigation accuracy increases.

The value assigned to P changes with each different implementation:

- NPS: P is equal to *d*. In other words, the robots transport the items directly to the home area from the items source. The downside of this approach is that each robot experiences a high error because the robots are travelling a long distance.

- SPS: The user assigns the same P to each robot in the swarm and P does not change over time. This value would be calibrated by the user to make the swarm exhibit the desired behaviour, in most cases to maximize item collection. The disadvantage of this approach is that it is not flexible. If the inherent error in the robots, the arena size or the robot platform changes the expected behaviour would be not the same. A further re-calibration might be required.

- CPS: P is discretized into a set of multiple smaller lengths. Each robot estimates the amount of time that it takes to transport an item with each partition in the set. The partition that provides the least amount of time has a bigger probability to be selected. The robots in order to learn the best distance that minimizes the time transporting item, the robots need to execute every distance in the set which can take a long time.

This thesis introduces a novel partitioning strategy referred to as the Dynamic Partitioning Strategy (DPS). In this strategy each robot learns P through the use of a penalty and reward mechanism.

The main findings in Chapter 4 describe how the ratio between the penalty and the reward (α) defines the frequency with which a robot finds an item (*collection ratio*). Selecting the appropriate *collection ratio* can provide a higher number of collected items than the other strategies. It is important to bear in mind the swarm size and the transaction time (the time required for the robots to exchange) in order to select the appropriate α.

Results shown in Chapter 5 portraits a discrepancy between error diversity and error uniformity. This chapter also highlights the importance of considering each result (heterogeneous and homogeneous errors) in order to have a complete comprehensive understanding of the swarm. Sensitivity analysis is performed to each parameter and output in each strategy to describe the influence of each parameter to each output.

Finally, Chapter 6 presents the contribution of each robot to the swarm when using each of the strategies. The robots converge to different solutions according to their inherent respective errors. In addition, the number of high error robots define the performance in the task. Experiments with physical robots validate the previous results; individual performance and final P changes from robot to robot.

7.2 Further Work

The penalty and reward mechanism is not only limited to change distance. The penalty and reward mechanism can be used for other tasks where a parameter needs to be changed as shown in Section 4.4. This thesis focus mainly to foraging tasks within the context of task partitioning. This work could be extended to other tasks such as item clustering and sorting, collaborative manipulation and swarm taxis. The penalty and reward mechanism could increase robustness against partial failures and reduce performance degradation in a similar way it does in foraging.

In the item clustering task robots collect items to a predefined region in a similar way as the conventional foraging task. The main difference is that the items are scattered across the environment and there are multiple collection points. An extension of this same task would be to have different categories of objects an the robots could have to take each item type to the appropriate collection point. The penalty and reward mechanism would help to increase item collection by rewarding the robots collecting the appropriate items to the collection point. These robots could increase the spatial radio exploration.

In the collaborative manipulation robots collaborate to transport heavy items where a single robot are unable to transport it by itself. In this case, the robots could be rewarded by the amount of distance they transport the item. Stronger robots would transport items for longer distances.

Swarm taxis is a variation of aggregation where robots move together towards a target in this case a beacon (Nembrini, 2005). A penalty and reward mechanism could be implemented where parameters would change according to the success or failure of the robot of losing connection with the swarm in a similar way as the aggregation algorithm shown in Section 4.4.

In addition, the discontinuity in the Pareto front found in the heterogeneous error can be further explored. This could potentially provide a measurement of degrees of robustness with respect to the number of high error robots. Therefore, it could be possible to identify the threshold of the number of high error robots for different swarm sizes that when overcome functionality is affected.

At this stage it is necessary to calibrate the value of α according the collection ratio desired by the user, as shown in Chapter 4. A potential extension can be implemented to optimize the performance of the robots with DPS. This might be achieved by combining CPS or DPS.

Communication between the robots can be implemented. This could lead to interesting results. In case of homogeneous error it could improve the convergence to solution speed. This is because each robot would share its partition length with each other. However, in case of heterogeneous error, the convergence would be slower due to the diversity in solutions of the swarm.

This research would be benefited with more experiments with physical robots. For instance, in this thesis a fixed swarm size is used for the experiments as shown in Chapter 6. Different swarm sizes would help to visualize the effects of having larger variety in the error of the robots in the swarm. In addition, for experiments with physical robots only the psi-swarm robot platform is used. Experiments with different robot platforms could show that the error diversity changes from each robot platform and DPS would work in spite of it.

Finally, the psi-swarm robot platform does not have a camera and external tracking system was used to aid robots as virtual camera. A fully decetralized robotic swarm system could be implemented where each robot has a camera. In theory, the results should be roughly the same.

Appendix A

Direct transference complementary figures

This appendix contains complementary figures of the Latin-hypercube analysis used in Chapter 4.

A.1 Introduction

In Section 4.3.3 the effect of using different reward and penalties are studied in more detail. In order to do this, a parameter is introduced referred as α. Only the results of α parameter were presented in Section 4.3.3. Results from the latin-hypercube analysis technique for *swarm size*, *d* and σ, are shown in this section.

A.2 Latin-hypercube analysis

Results from the parameters *swarm size,* distance home area - items source (*d*) and *dead reckoning noise* (σ).

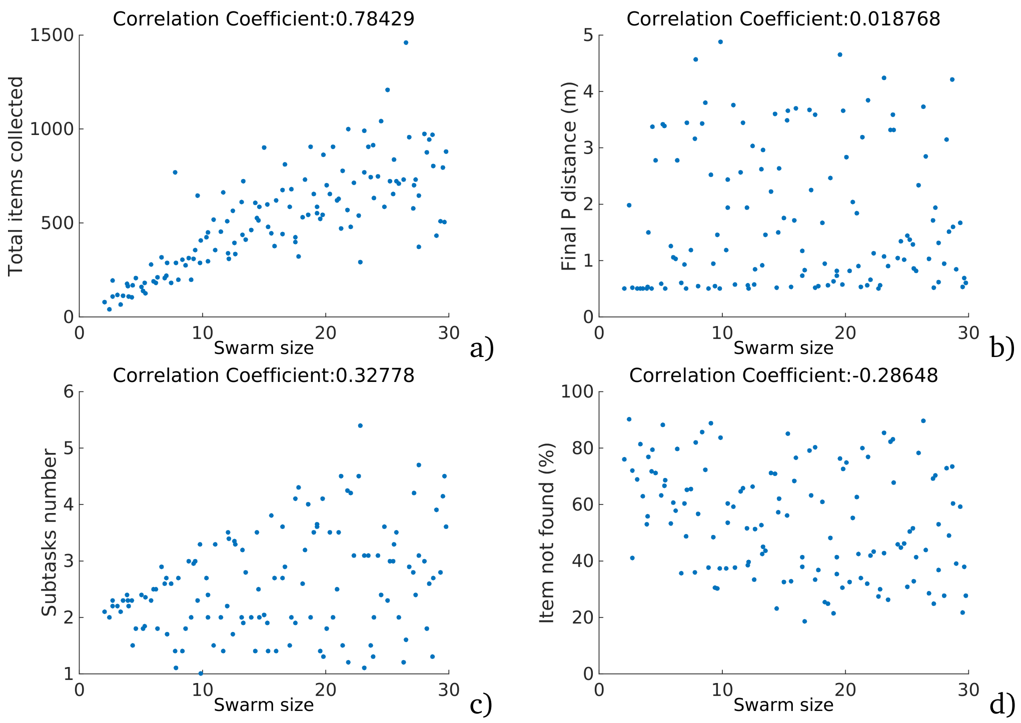


Figure A.1: Latin-hypercube analysis for the *swarm size* parameter. The outputs shown are total items collected, final P distance, subtasks number and item not found percentage (a, b, c and respectively).

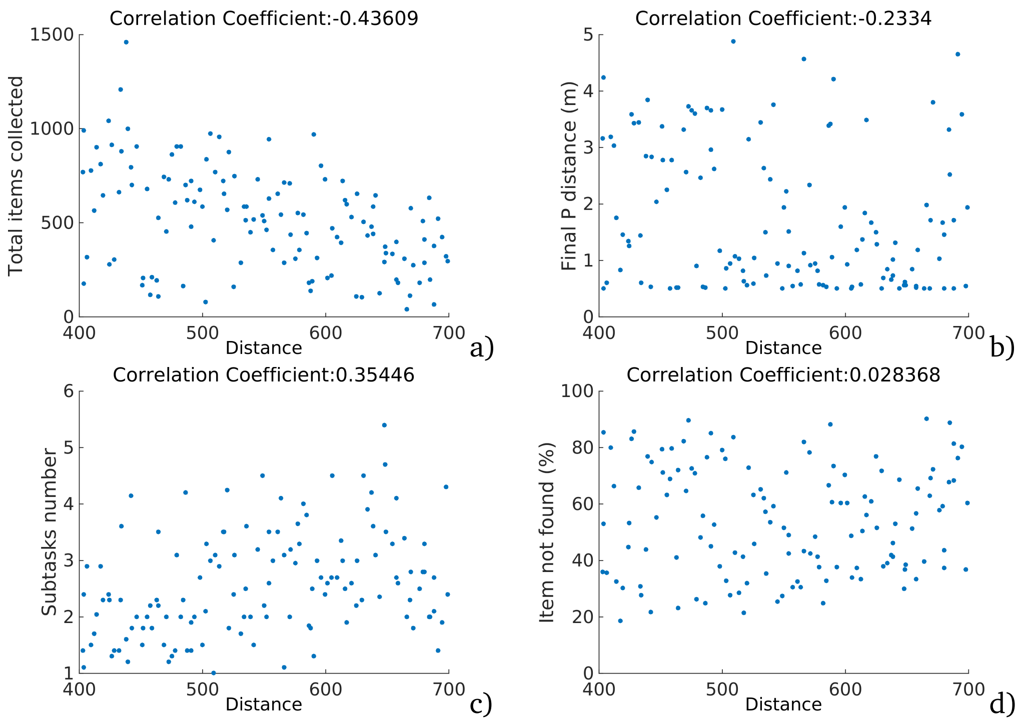


Figure A.2: Latin-hypercube analysis for the *d* parameter. The outputs shown are total items collected, final P distance, subtasks number and item not found percentage (a, b, c and respectively).

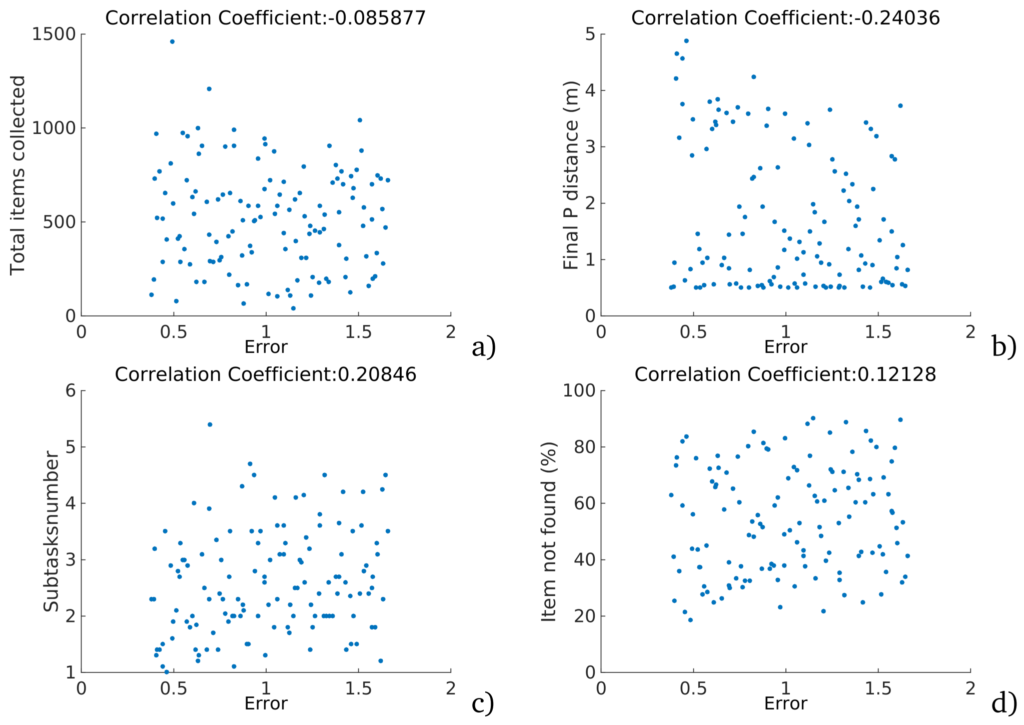


Figure A.3: Latin-hypercube analysis for the σ parameter. The outputs shown are total items collected, final P distance, subtasks number and item not found percentage (a, b, c and respectively).

Appendix B

Ensemble generation

This appendix describes the training and generation of each emulator used in Chapter 5. In addition, complementary results from the latin-hypercube sampling, eFast, easyABC and NSGA-II are shown in this appendix.

B.1 Introduction

Ensemble generation is a technique provided by the statistical tool Spartan (Alden et al., 2018). Ensembles are composed of different emulations generated with different machine learning techniques. The data set used to train each ensemble is generated from latin-hypercube analysis. By using an emulator it is possible to reduce time and computational resources inherent in simulations due to the large number of replicates, length of the simulations and computation expenses. Enriched analysis such as Approximate Bayesian Computation and Multi Objective Evolutinary Computation algorithms can be performed on emulations with a relatively low computation expenses in a small time.

The machine learning techniques used to train each ensemble are: neural network, random forest, support vector machine, gaussian process model and general linear model.

The statistical tools used to explore each strategy are LHA (Mckay et al., 1998), efast (Saltelli and Bollardo, 1998), easyABC (Blum et al., 2010) and NSGA-II (Deb et al., 2002).

A total of 8 emulations are generated for each of the four strategies (NPS, SPS, CPS and DPS) for heterogeneous and homogeneous error.

B.2 Dataset

Dataset used to train emulations for Chapter 5 was generated from a latin-hypercube analysis performed to each strategy and each error type. Parameters and outputs used for each strategy

Table B.1: Parameters and outputs used for each strategy

Parameters			Outputs	
Strategy	Name	Interval	Name	Interval
NPS	*Swarm size*	[2-14]	*total items collected*	[0 max]
	d	[0.5-2.0 m]	*explore ratio*	[0 1]
SPS	*Swarm size*	[2-14]	*collection ratio*	[0 1]
	d	[0.5-2.0 m]		
	P	[0.2-2.0 m]		
DPS	*Swarm size*	[2-14]		
	d	[0.5-2.0 m]		
	α	[0-1]		
CPS	*Swarm size*	[2-14]		
	d	[0.5-2.0 m]		
	ϵ	[0-1]		
	Memory factor	[0-1]		

are shown in Table B.1.

B.3 Evaluation

In order to measure the accuracy of the emulator, the root-mean-square-deviation(RMSD) is used to measure the differences between the values used to train the ensemble and the value predicted. Results are shown next.

B.4 Complementary figures

B.4.1 Latin-hypercube analysis technique

Complementary results from the latin-hypercube analysis used for the experiments in Section 5.4.

B.4.2 The easyABC technique

Complementary results from the easyABC technique used for the experiments in Section 5.4.

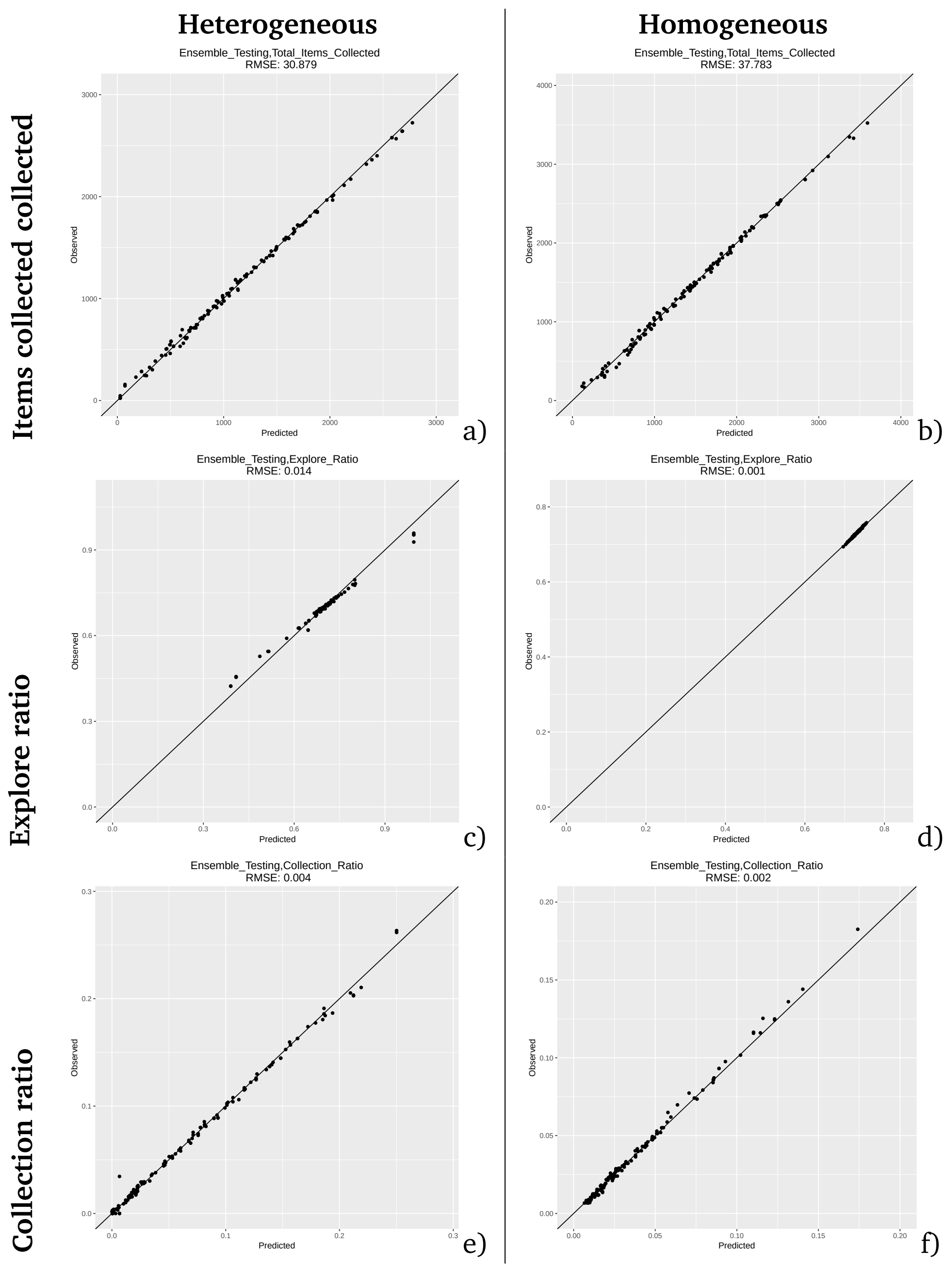


Figure B.1: Testing sets for NPS for heterogeneous and homogeneous error (left and right column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

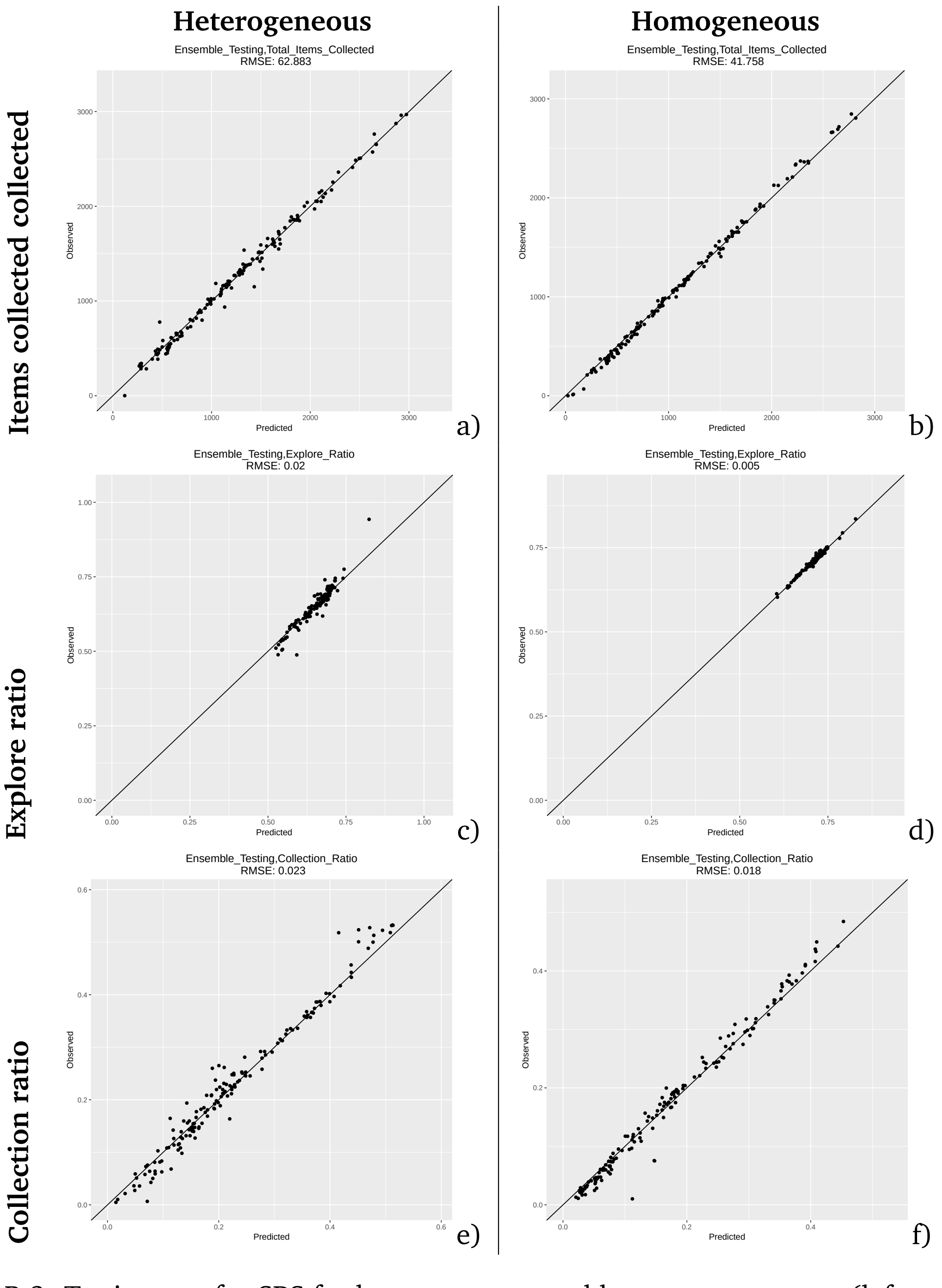


Figure B.2: Testing sets for SPS for heterogeneous and homogeneous error (left and right column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

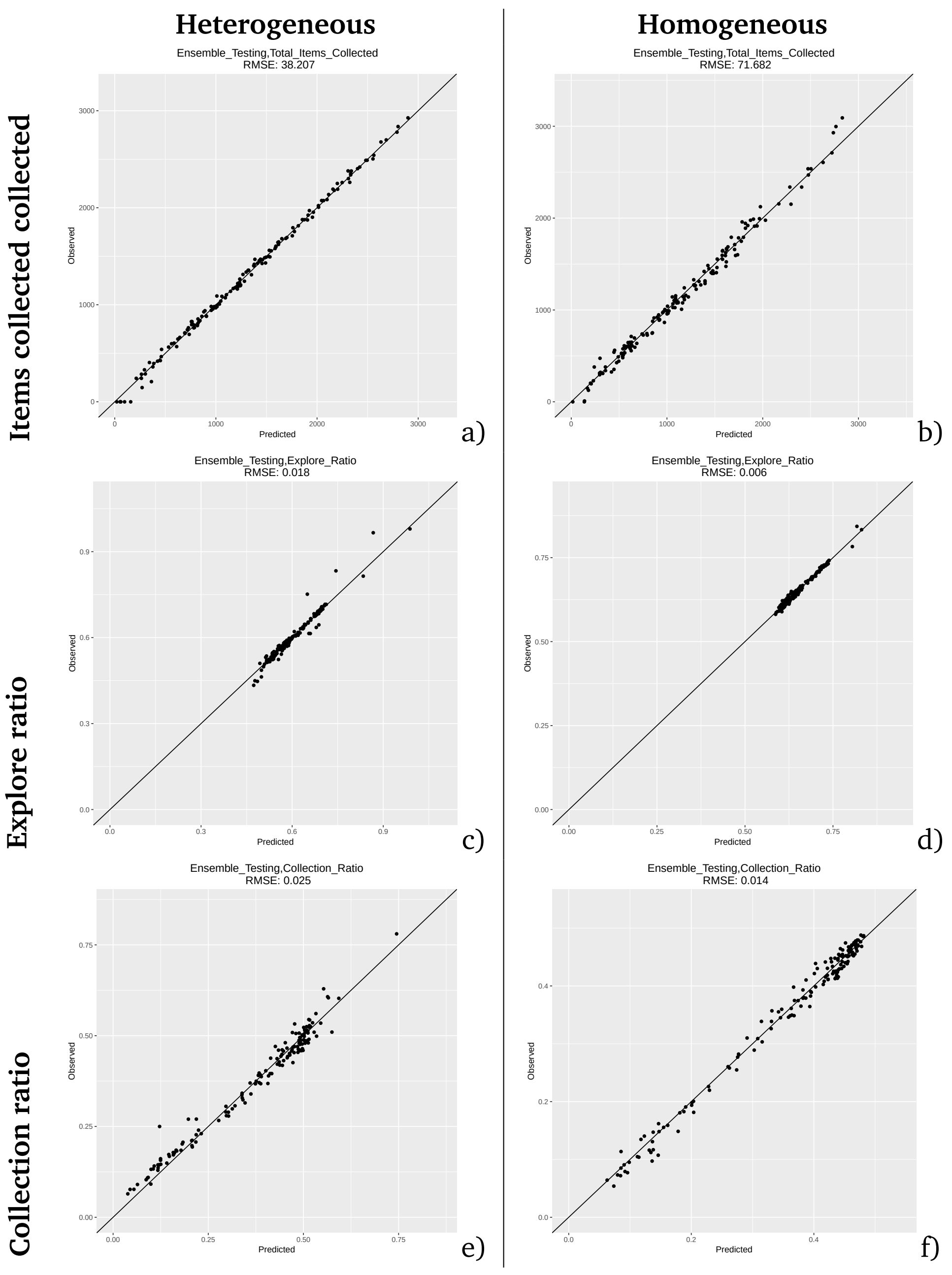


Figure B.3: Testing sets for DPS for heterogeneous and homogeneous error (left and right column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

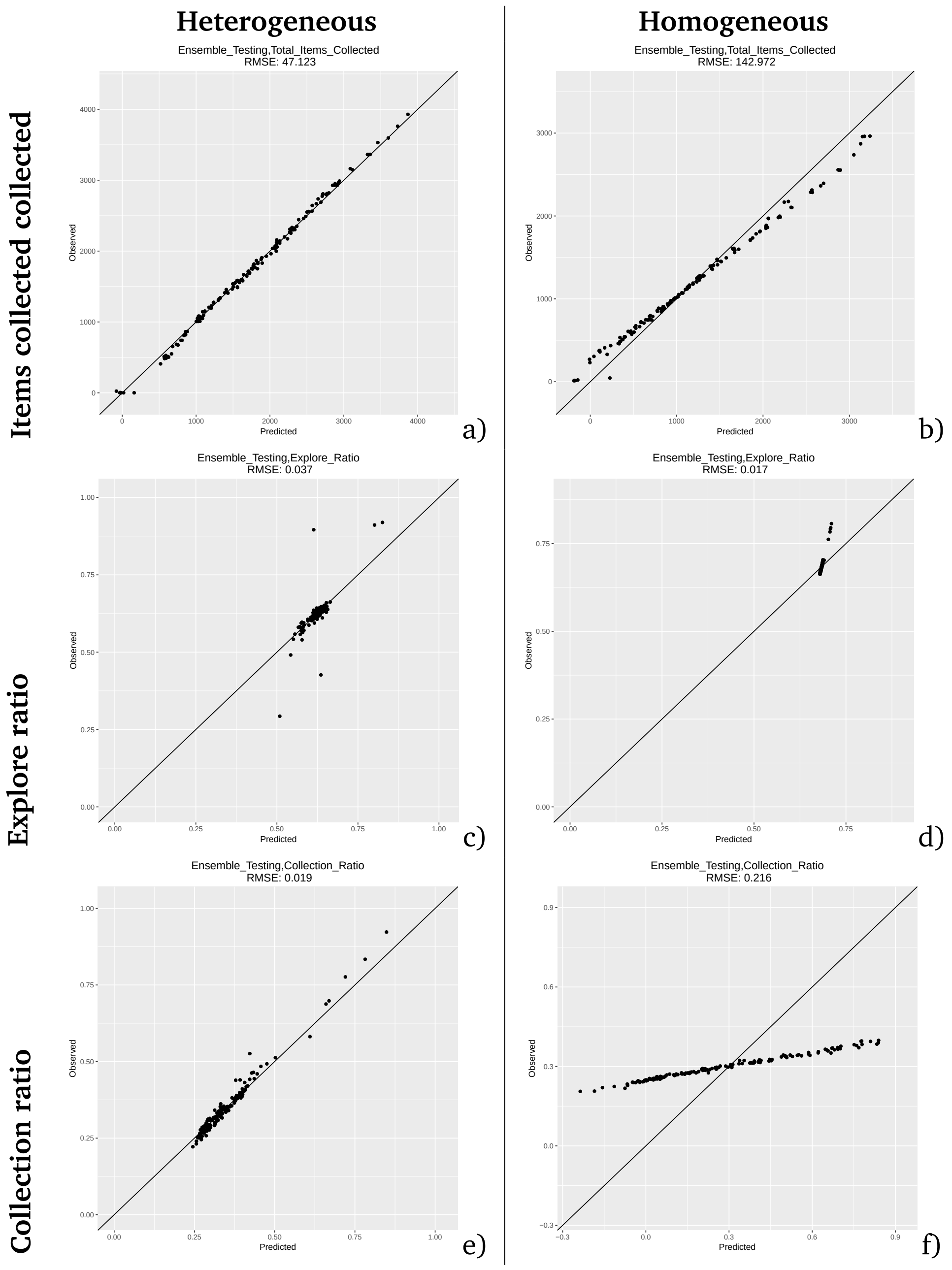


Figure B.4: Testing sets for CPS for heterogeneous and homogeneous error (left and right column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

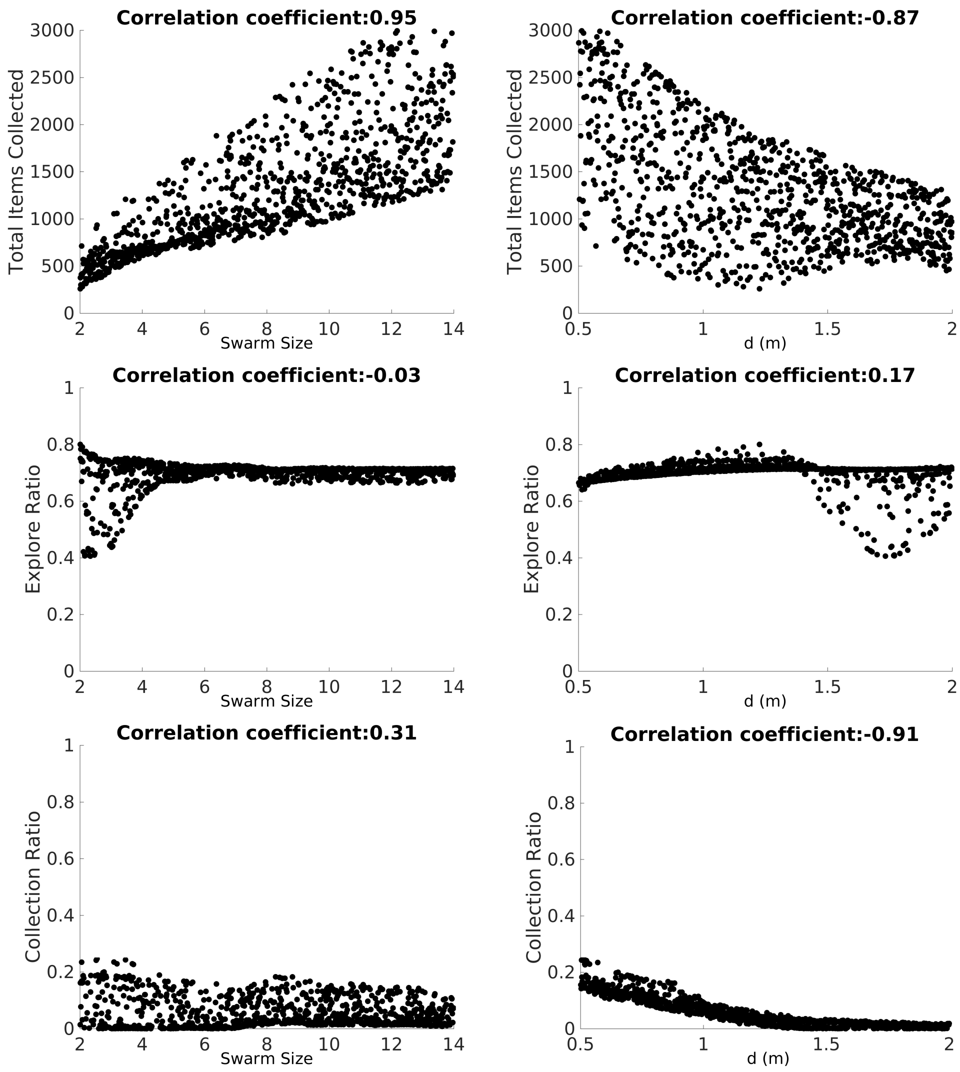


Figure B.5: LHA for NPS for heterogeneous error. Parameters: swarm size and d (left and right column). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

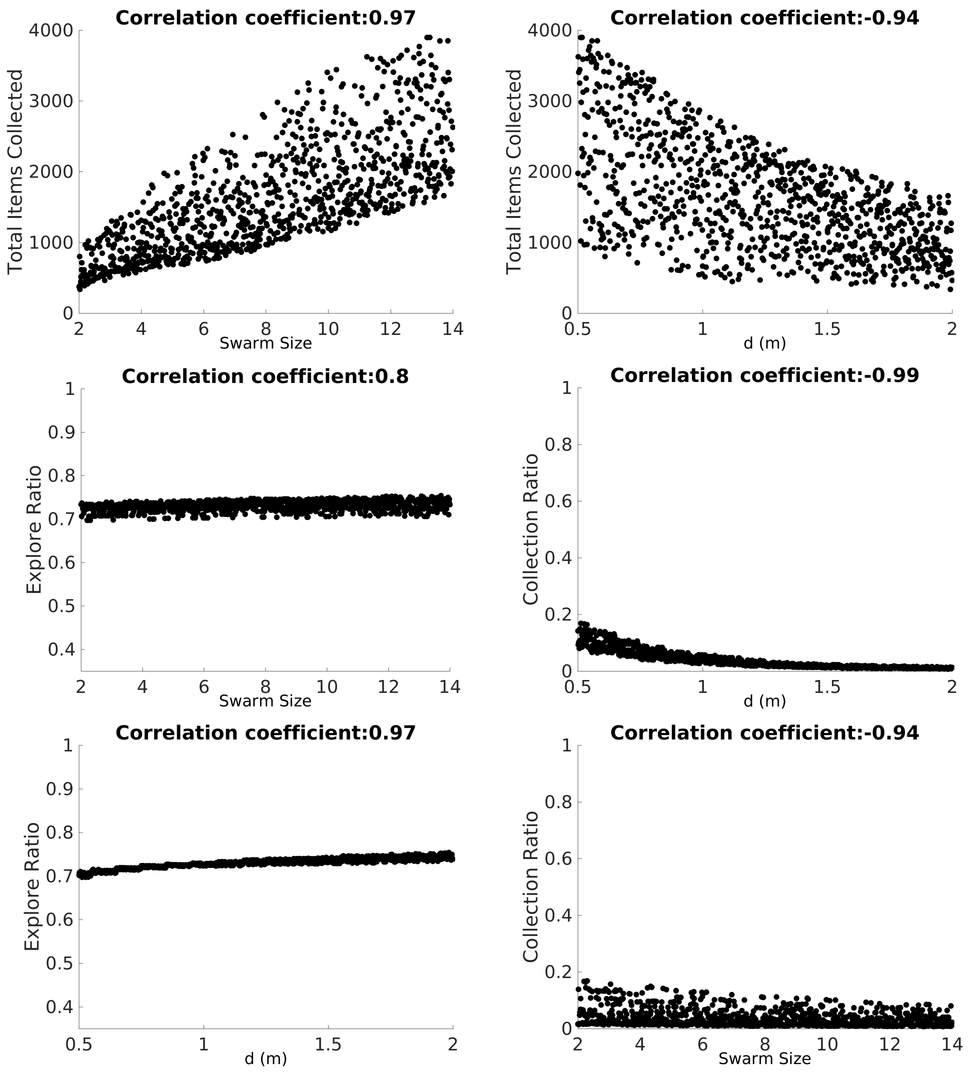


Figure B.6: LHA for NPS for homogeneous error. Parameters: swarm size and d (left and right column). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

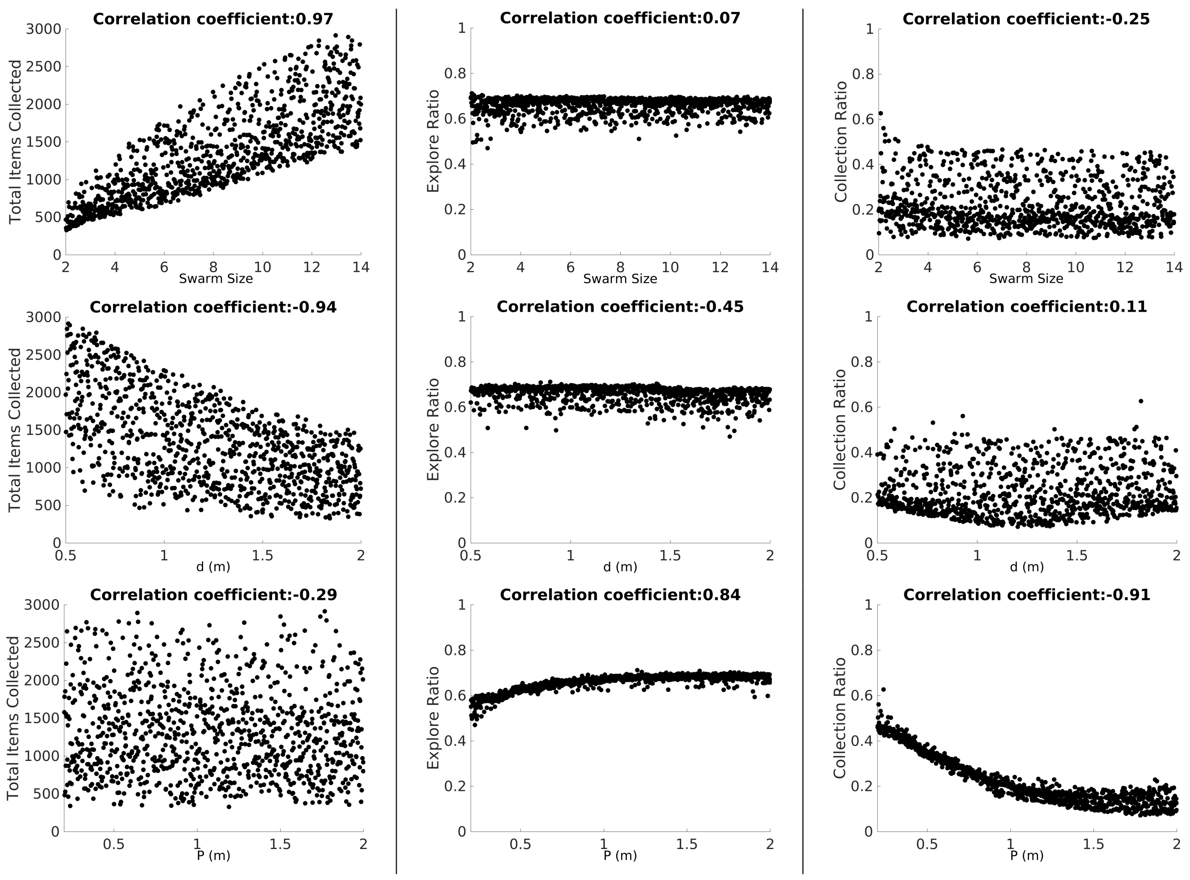


Figure B.7: LHA for SPS for heterogeneous error. Parameters: swarm size, d and P (first, second and third row respectively). Outputs: total items collected, explore ratio and collection ratio (left, center and right column respectively).

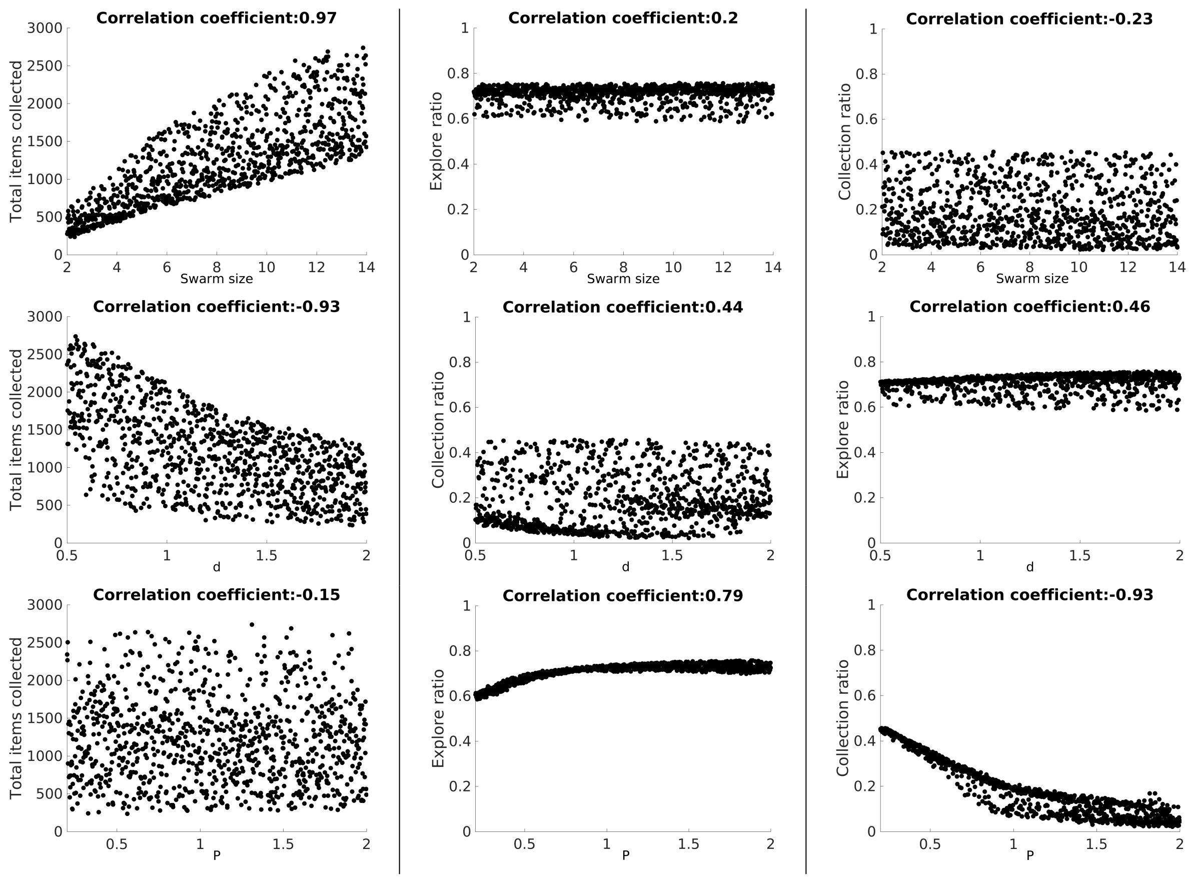


Figure B.8: LHA for SPS for homogeneous error. Parameters: swarm size, d and P (first, second and third row respectively). Outputs: total items collected, explore ratio and collection ratio (left, center and right column respectively).

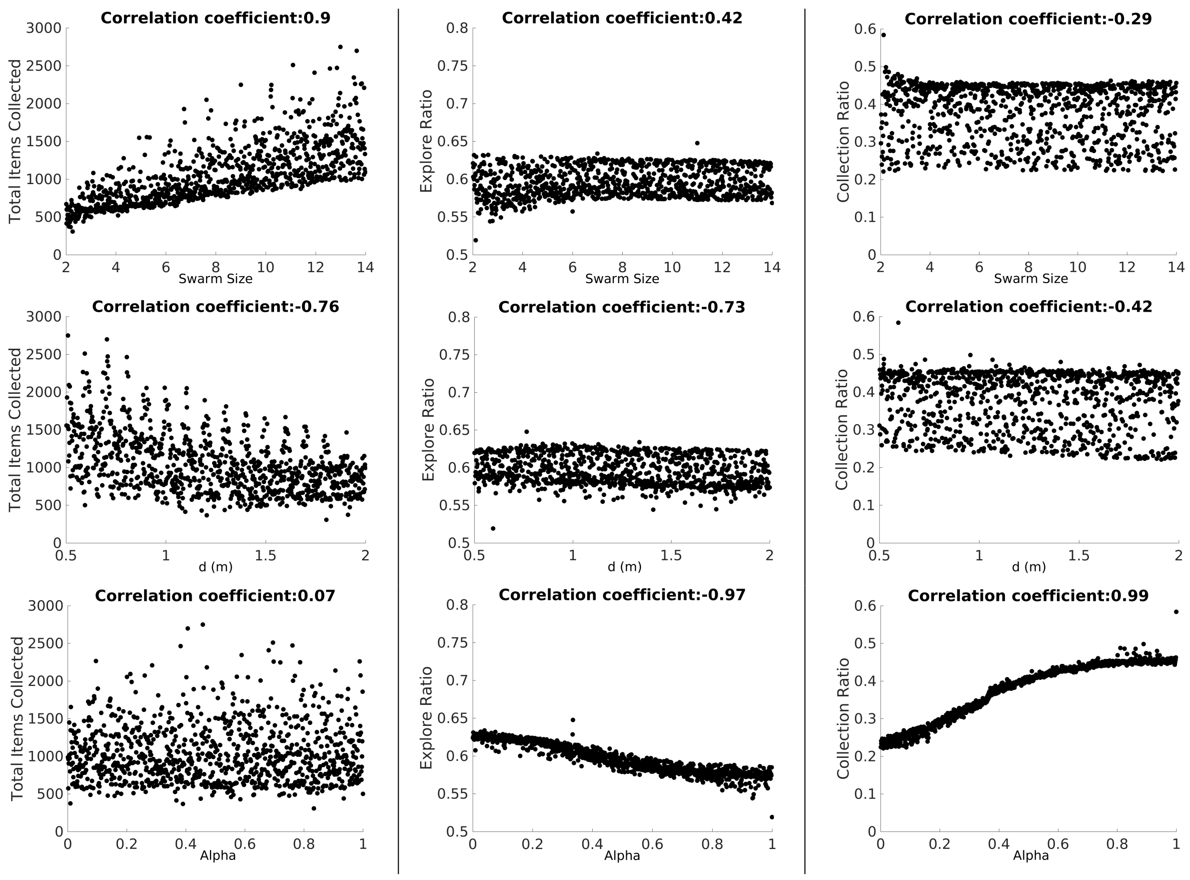


Figure B.9: LHA for DPS for heterogeneous error. Parameters: swarm size, d and α (first, second and third row respectively). Outputs: total items collected, explore ratio and collection ratio (left, center and right column respectively).

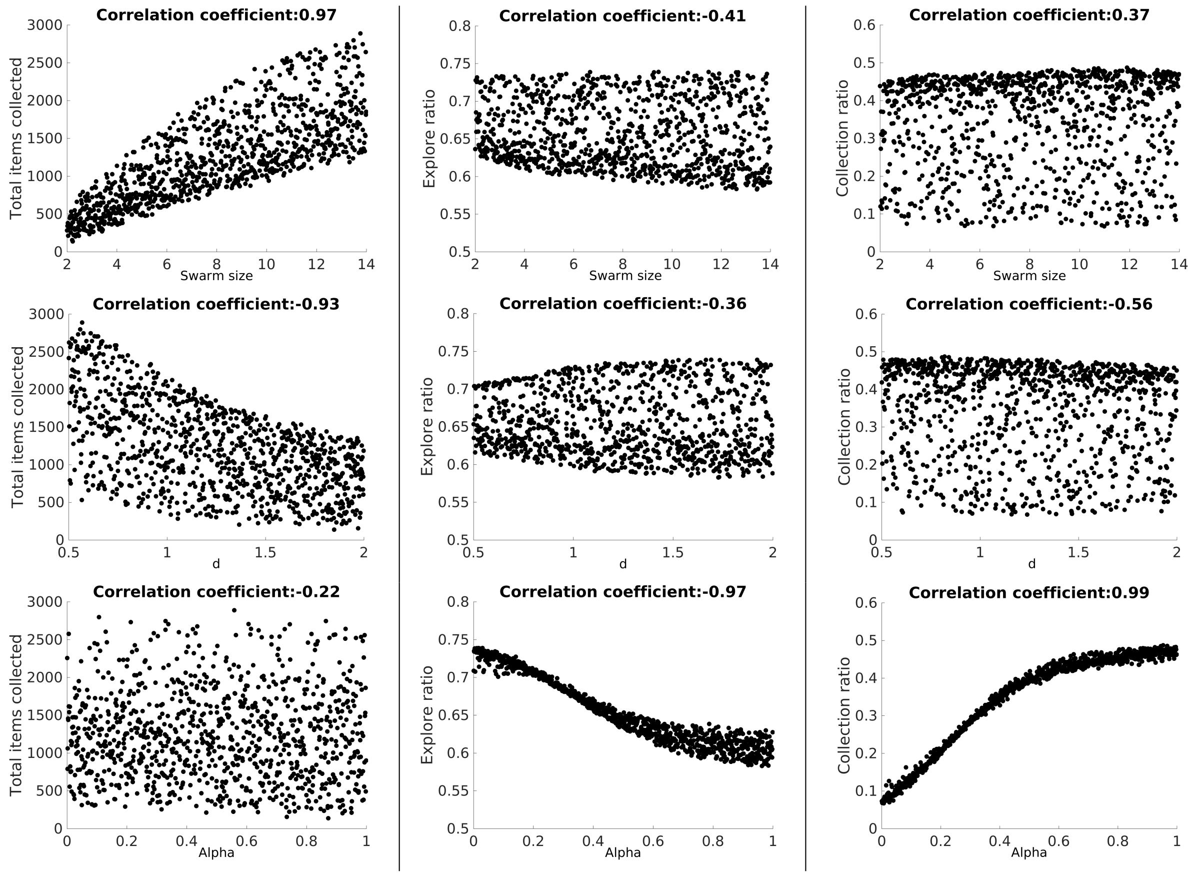


Figure B.10: LHA for DPS for homogeneous error. Parameters: swarm size, d and α (first, second and third row respectively). Outputs: total items collected, explore ratio and collection ratio (left, center and right column respectively).

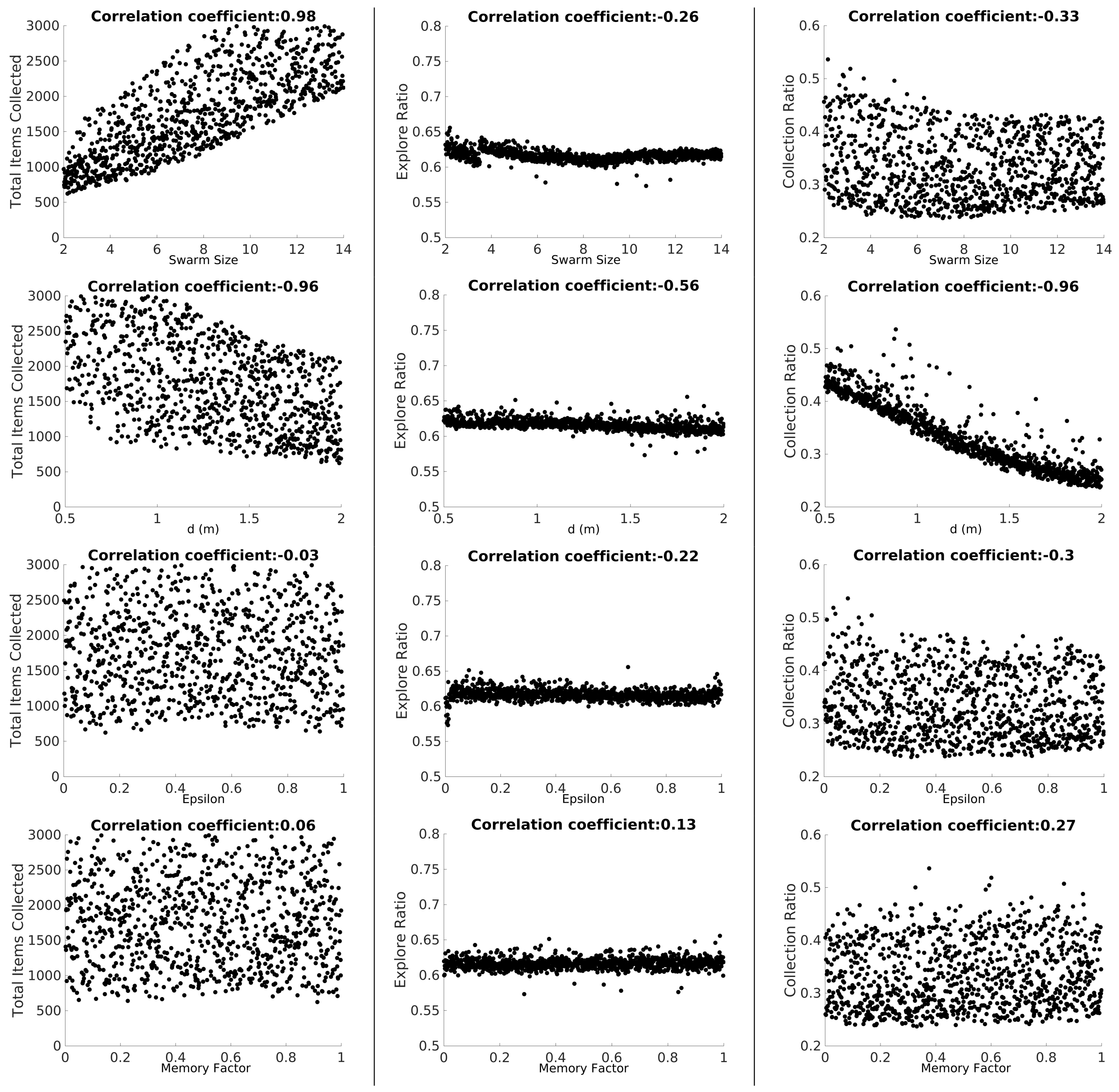


Figure B.11: LHA for CPS for heterogeneous error. Parameters: swarm size, d and ϵ and memory factor (first, second, third and fourth row respectively). Outputs: total items collected, explore ratio and collection ratio (left, center and right column respectively).

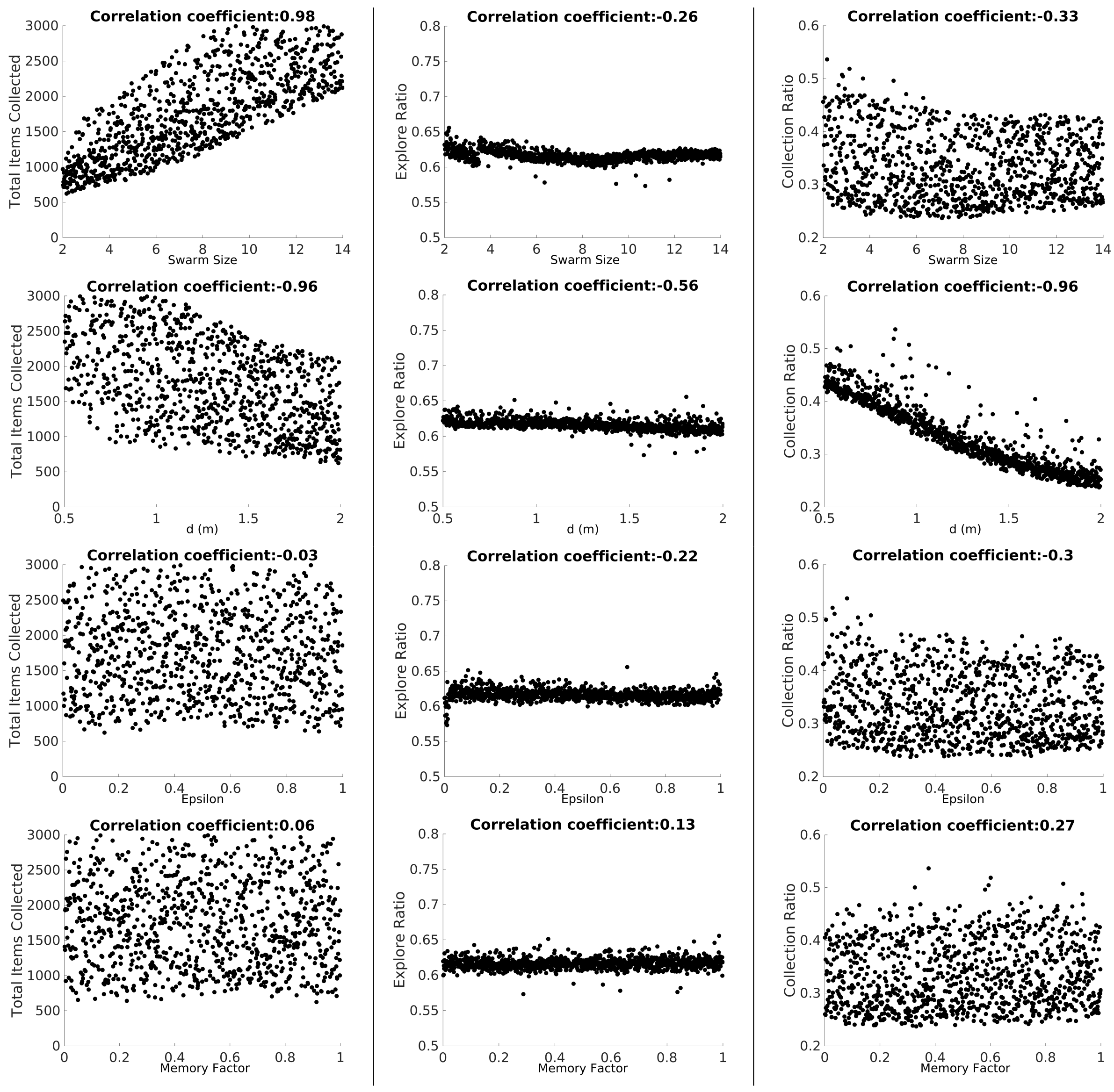


Figure B.12: LHA for CPS for homogeneous error. Parameters: swarm size, d and ϵ and memory factor (first, second, third and fourth row respectively). Outputs: total items collected, explore ratio and collection ratio (left, center and right column respectively).

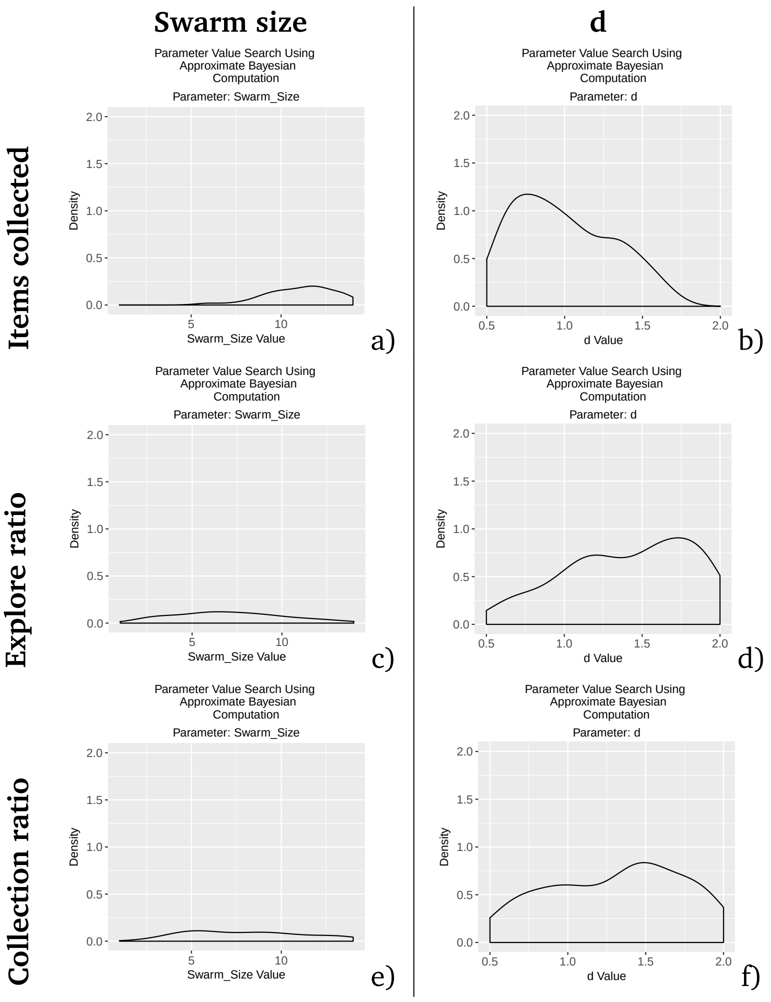


Figure B.13: easyABC results for NPS with heterogeneous. Parameters: swarm size and d (first and second column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

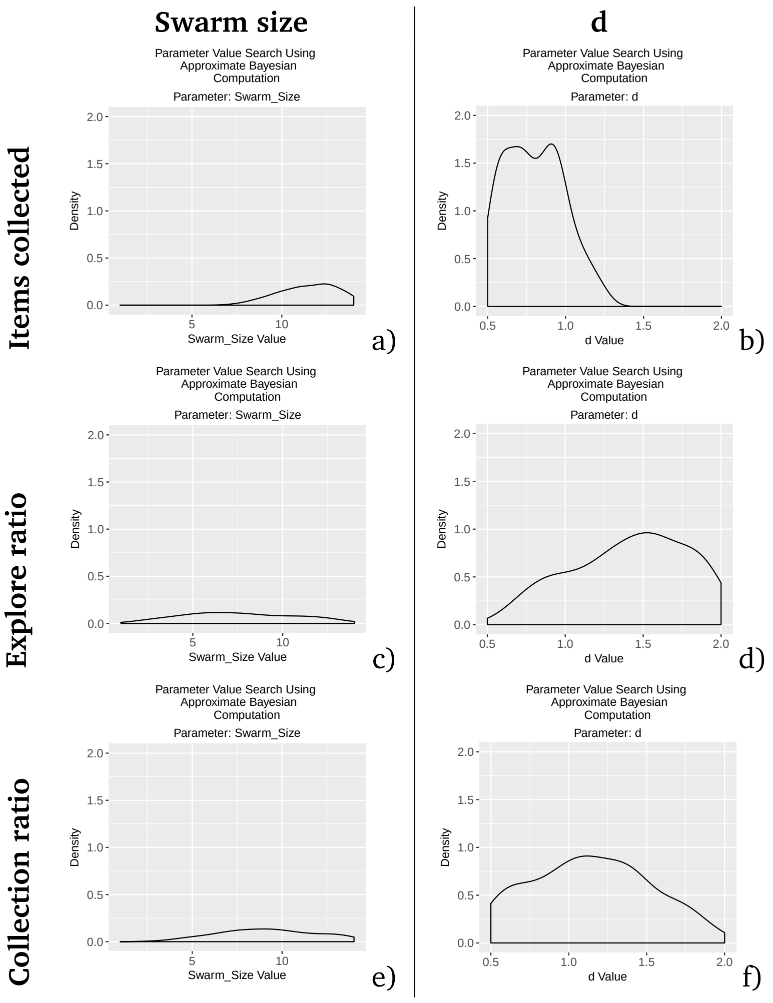


Figure B.14: easyABC results for NPS with homogeneous error. Parameters: swarm size and d (first and second column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

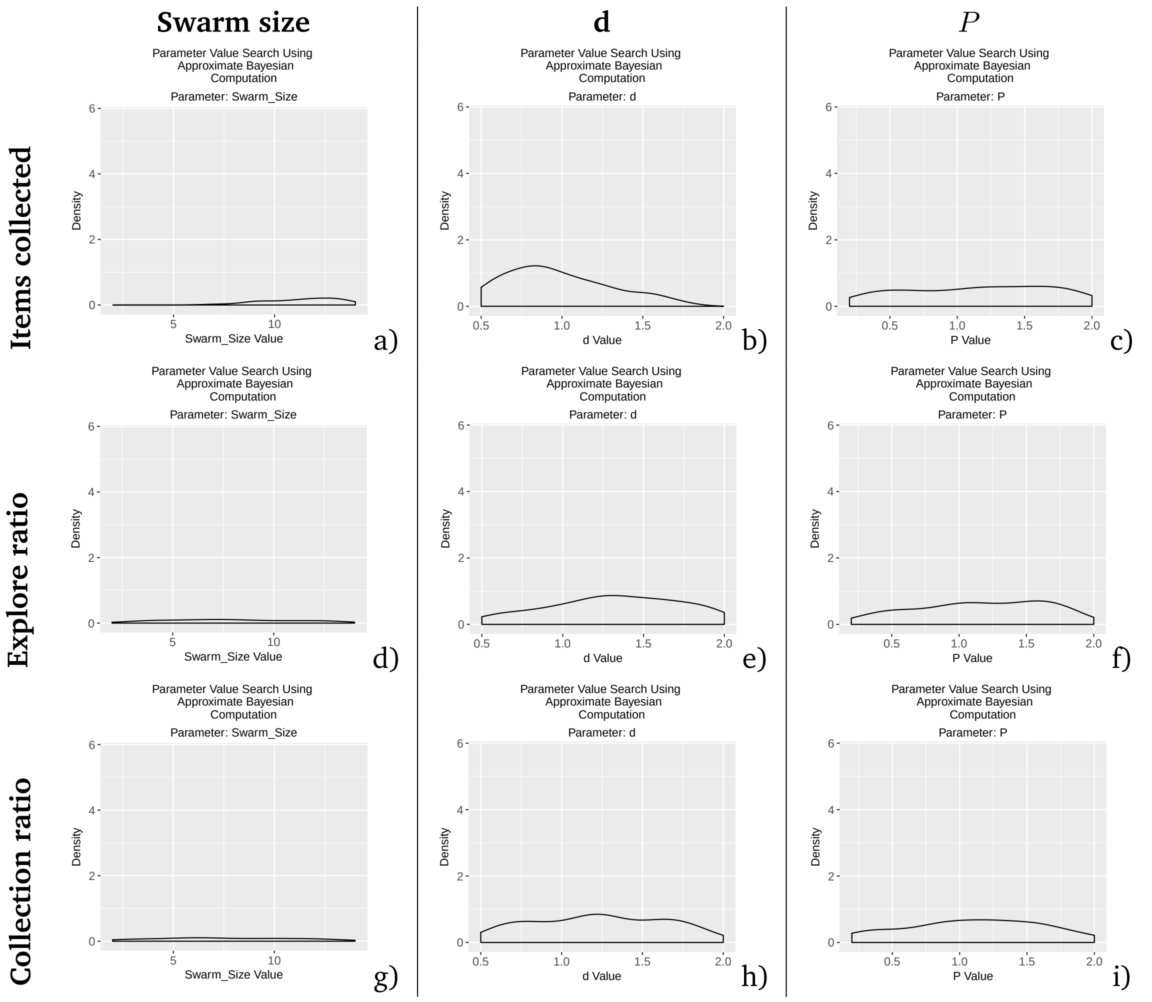


Figure B.15: easyABC results for SPS with heterogeneous error. Parameters: swarm size, d and P (first, second and third column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

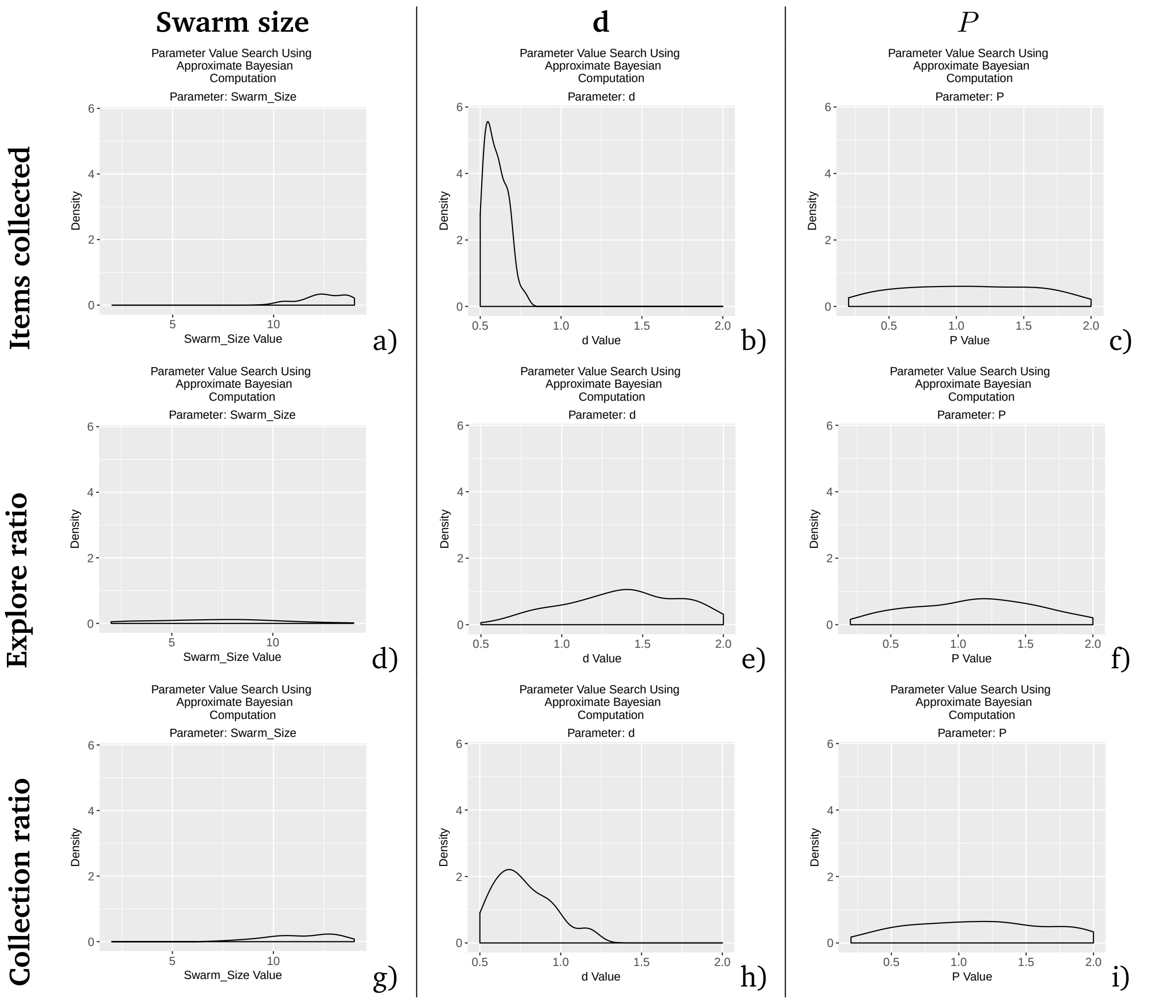


Figure B.16: easyABC results for SPS with homogeneous error. Parameters: swarm size, d and P (first, second and third column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

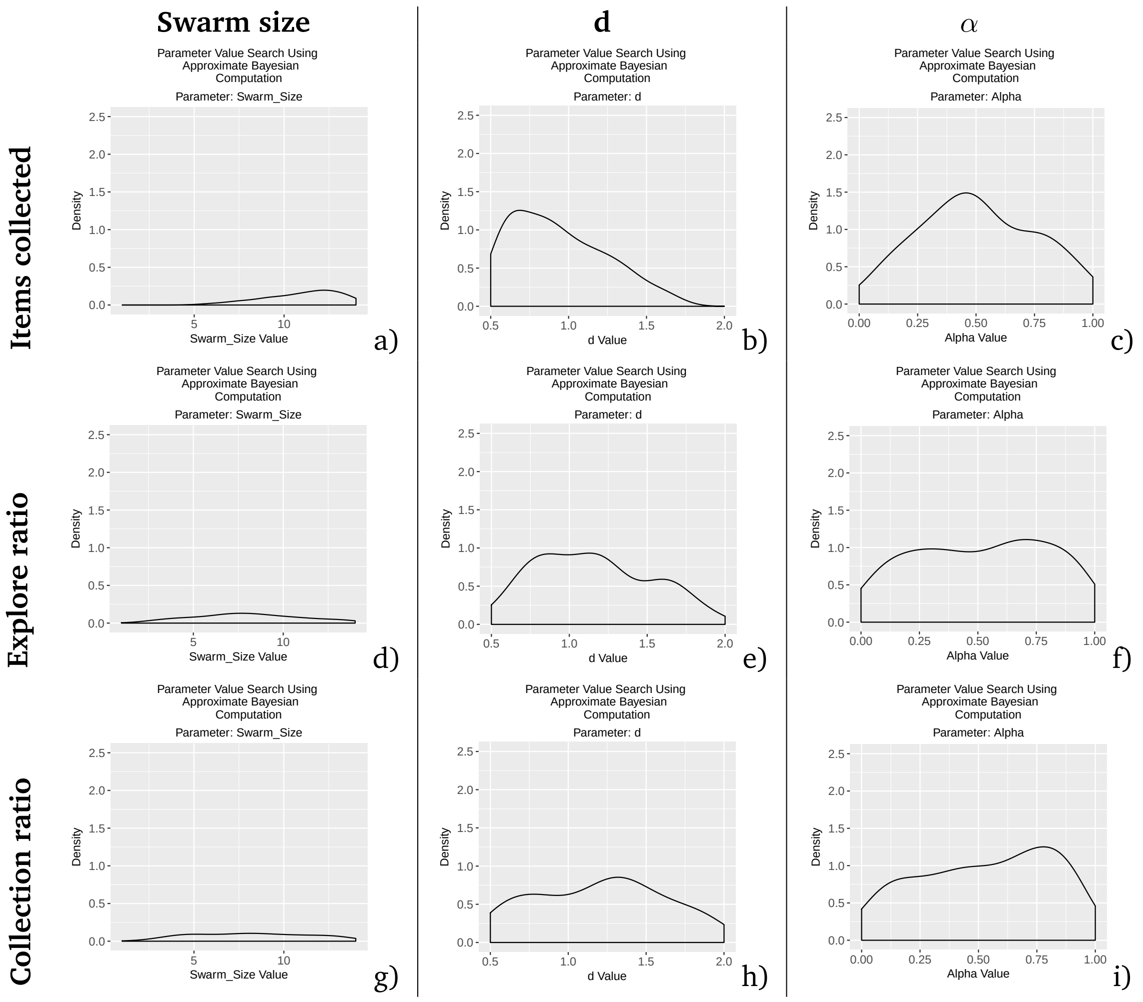


Figure B.17: easyABC results for DPS with heterogeneous error. Parameters: swarm size, d and α (first, second and third column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

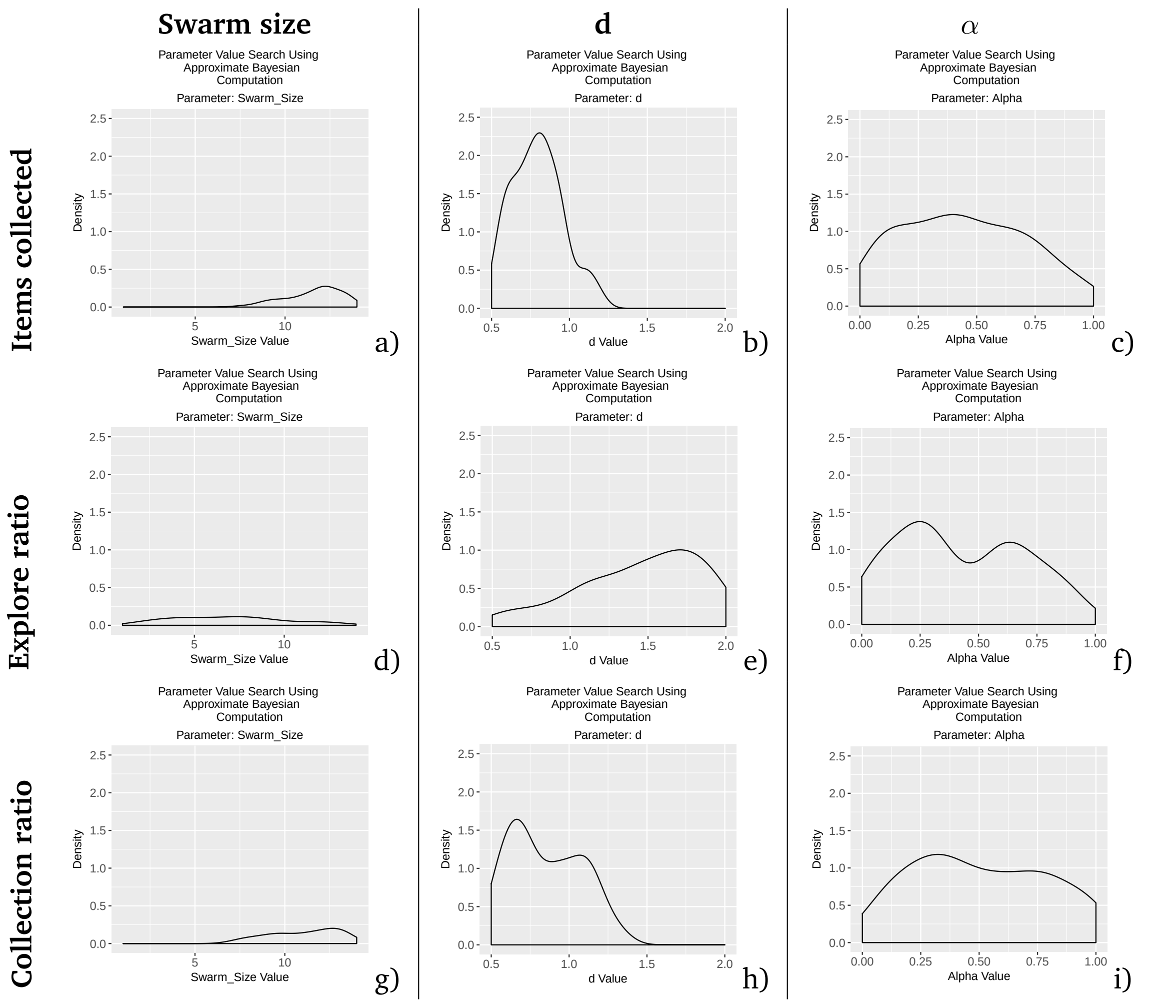


Figure B.18: easyABC results for DPS with homogeneous error. Parameters: swarm size, d and α (first, second and third column respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third row respectively).

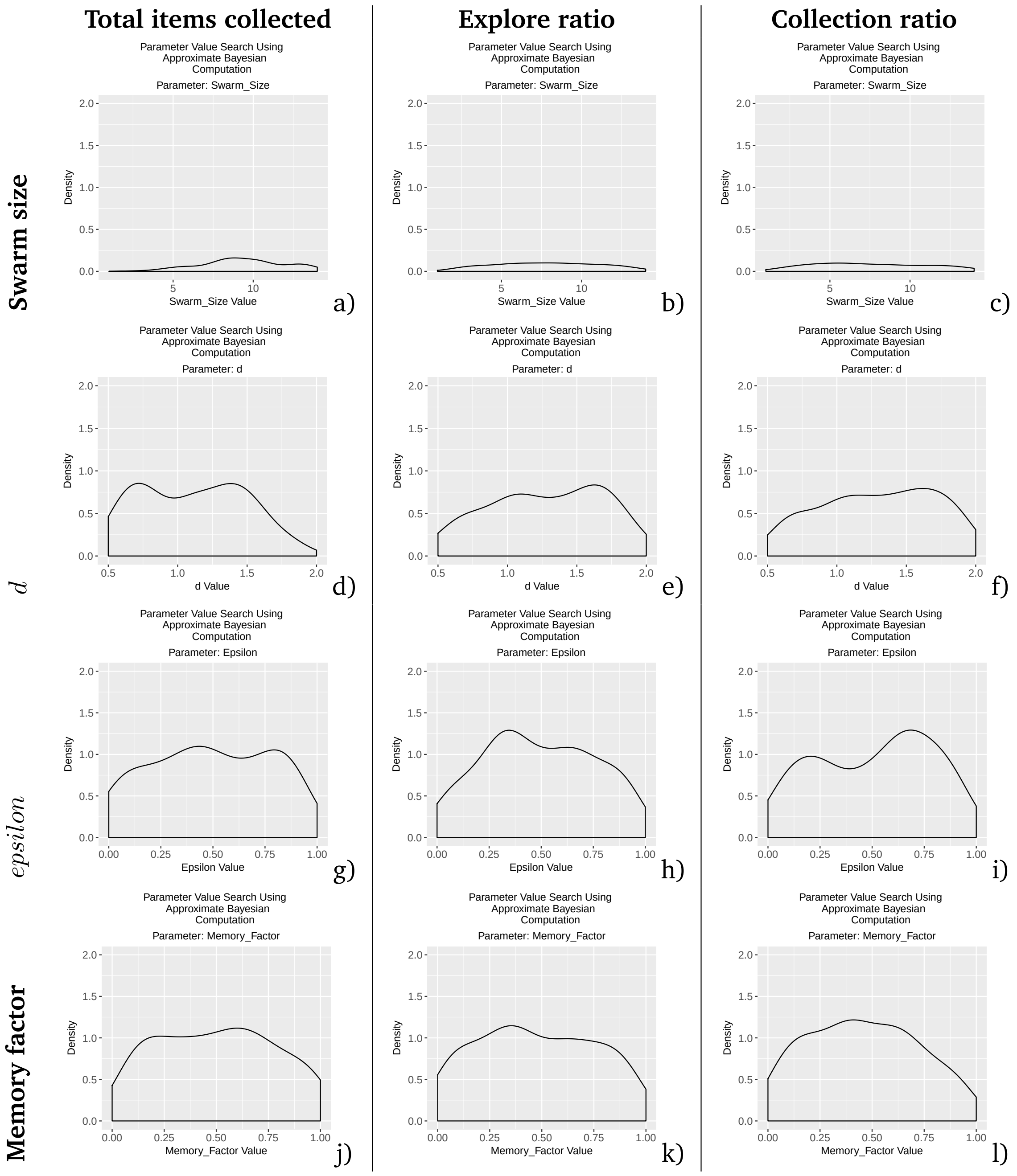


Figure B.19: easyABC results for CPS with heterogeneous error. Parameters: swarm size, d, ϵ and memory factor (first, second, third and fourth row respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third column respectively).

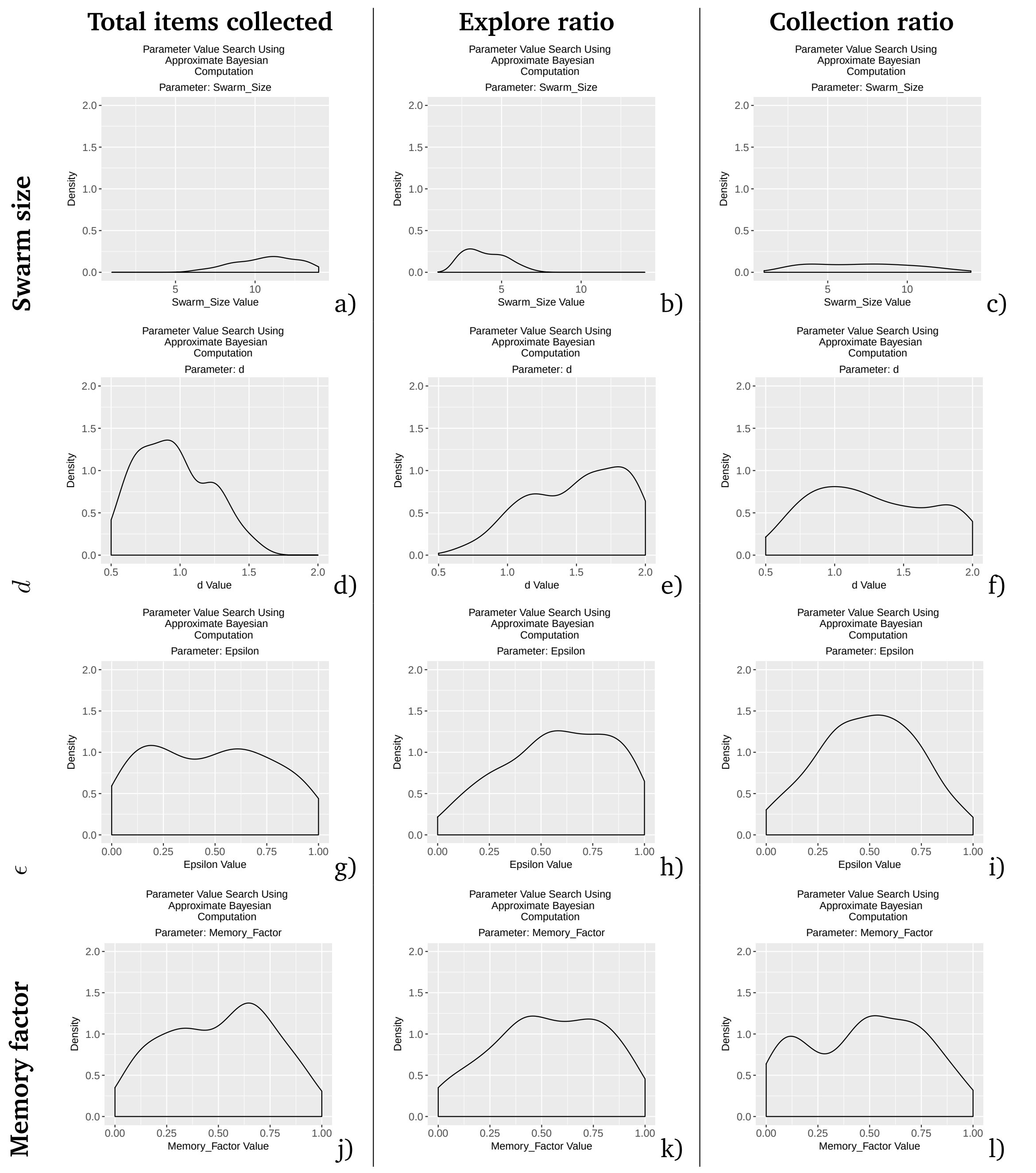


Figure B.20: easyABC results for CPS with homogeneous error. Parameters: swarm size, d, ϵ and memory factor (first, second, third and fourth row respectively). Outputs: total items collected, explore ratio and collection ratio (first, second and third column respectively).

Appendix C

Bibliography

Bibliography

Levent Bayındır. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

Jennifer Carlson and Robin R. Murphy. How UGVs physically fail in the field. *IEEE Transactions on Robotics*, 21(3):423–437, 2005. ISSN 15523098. doi: 10.1109/TRO.2004.838027.

Johann Borenstein and Liqiang Feng. Measurement and Correction of Systematic Odometry Errors in Mobile Robots. *Transaction on Robotics and Automation*, 12(6):869–880, 1996.

Giovanni Pini, Arne Brutschy, Alexander Scheidler, and et al. Task Partitioning in a Robot Swarm: Object Retrieval as Sequence of Subtasks with Direct Object Transfer. *Artificial Life*, 20(3):291–317, 2014. ISSN 10645462. doi: 10.1162/ARTL.

G. Pini, a. Brutschy, C. Pinciroli, M. Dorigo, and M. Birattari. Autonomous task partitioning in robot foraging: an approach based on cost estimation. *Adaptive Behavior*, 21(2):118–136, 2013. ISSN 1059-7123. doi: 10.1177/1059712313484771.

Jan Carlo Barca and Y. Ahmet Sekercioglu. Swarm robotics reviewed. *Robotica*, 31(3):1–15, 2012. ISSN 0263-5747. doi: 10.1017/S026357471200032X.

Alan F.T. Winfield and Julien Nembrini. Safety in numbers: fault-tolerance in robot swarms, 2006. ISSN 1746-6172.

Erol Sahin. Swarm Robotics : From Sources of Inspiration. *Swarm robotics workshop: state-of-the-art survey*, pages 10–20, 2005. doi: 10.1007/978-3-540-30552-1-2.

Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, jan 2013. ISSN 1935-3812. doi: 10.1007/s11721-012-0075-2. URL `http://link.springer.com/10.1007/s11721-012-0075-2`.

Jan Dyre Bjerknes and Alan F T Winfield. On fault tolerance and scalability of swarm robotic systems. *Springer Tracts in Advanced Robotics*, 83 STAR:431–444, 2012. ISSN 16107438. doi: 10.1007/978-3-642-32723-0_31.

Huikeng Lau, Iain Bate, and Jon Timmis. Immune-Inspired Error Detection for Multiple Faulty Robots in Swarm Robotics. *ECAL 2013*, pages 846–853, 2013. doi: 10.7551/978-0-262-31709-2.

Alan F T Winfield, Christopher J Harper, and Julien Nembrini. Towards Dependable Swarms and a New Discipline of Swarm Engineering Case Study : Swarm Containment. *Robotics*, pages 126–142, 2005. doi: 10.1007/978-3-540-30552-1{_}11.

Anders Lyhne Christensen, Rehan O Grady, and Marco Dorigo. From Fireflies to Fault-Tolerant Swarms of Robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009.

Lynne E. Parker. L-ALLIANCE: Task-oriented multi-robot learning in behavior-based systems. *Advanced Robotics*, 11(4):305–322, 1996. ISSN 0169-1864. doi: 10.1163/156855397X00344. URL `http://www.tandfonline.com/doi/abs/10.1163/156855397X00344`.

Danesh Tarapore, Pedro U Lima, Jorge Carneiro, and Anders Lyhne Christensen. To err is robotic, to tolerate immunological: fault detection in multirobot systems. *Bioinspiration & Biomimetics*, 10(1):016014, 2015. ISSN 1748-3190. doi: 10.1088/1748-3190/10/1/016014.

Xingyan Li and Lynne E Parker. Sensor Analysis for Fault Detection in Tightly-Coupled Multi-Robot Team Tasks. In *Proceedings - IEEE International Conference on Robotics and Automation*, number April, pages 3269–3276, 2007. ISBN 9781424427895. doi: 10.1109/ROBOT.2009.5152389.

Lynne E Parker and Balajee Kannan. Adaptive causal models for fault diagnosis and recovery in multi-robot teams. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2703–2710, 2006. ISBN 142440259X. doi: 10.1109/IROS.2006.281993.

James O'Keeffe, Danesh Tarapore, Alan G Millard, and Jon Timmis. Towards fault diagnosis in robot swarms: An online behaviour characterisation approach. pages 393–407, 2017.

Raja Humza, Oliver Scholz, Maizura Mokhtar, Jon Timmis, and Andy Tyrrell. Towards energy homeostasis in an autonomous self-reconfigurable modular robotic organism. *Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns, ComputationWorld 2009*, pages 21–26, 2009. doi: 10.1109/ComputationWorld.2009.83.

M.T. Long, R.R. Murphy, and L.E. Parker. Distributed multi-agent diagnosis and recovery from sensor failures. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 3(October):2506–2513, 2003. doi: 10.1109/IROS.2003.1249246.

Jon Timmis, Andy Tyrrell, M Mokhtar, Amelia Ismail, N Owens, and Ran Bi. An Artificial Immune System for Robot Organisms. *Symbiotic Multi-Robot Organisms: Reliability, Adaptability*, pages 279–302, 2010.

Shervin Nouyan, Alexandre Campo, and Marco Dorigo. Path formation in a robot swarm. *Swarm Intelligence*, 2(1):1–23, dec 2007. ISSN 1935-3812. doi: 10.1007/s11721-007-0009-6. URL `http://link.springer.com/10.1007/s11721-007-0009-6`.

Roderich Groß, Shervin Nouyan, Michael Bonani, Francesco Mondada, and Marco Dorigo. Division of Labour in Self-organised Groups. *SAB 2008*, pages 426–436, 2008.

Shervin Nouyan, Roderich Groß, Michael Bonani, Francisco Mondada, and Marco Dorigo. Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711, 2009. ISSN 1089778X. doi: 10.1109/TEVC.2008.2011746.

Thomas Schmickl, Ronald Thenius, Christoph Möslinger, Jon Timmis, Andy Tyrrell, Mark Read, James Hilder, Jose Halloy, Alexandre Campo, Cesare Stefanini, Luigi Manfredi, Stefano Orofino, Serge Kernbach, Tobias Dipper, and Donny Sutantyo. CoCoRo - The self-aware underwater swarm. *Proceedings - 2011 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2011*, 1:120–126, 2011. doi: 10.1109/SASOW.2011.11.

Goss Simon, Serge Aron, Jean-Lous Deneubourg, and Jacques Pasteels. Self-organized Shortcuts in the Argentine Ant. *Springer-Verlag*, 514(1959):579–581, 1989.

David Payton, Mike Daily, Regina Estowski, Mike Howard, and Craig Lee. Pheromone robotics. *Autonomous Robots*, 11:319–324, 2001. ISSN 09295593. doi: 10.1023/A:1012411712038.

David Payton, Regina Estkowski, and Mike Howard. Pheromone Robotics and the Logic of Virtual Pheromones. *Swarm Robotics*, pages 45–57, 2005.

David Payton, Regina Estkowski, and Mike Howard. Compound behaviors in pheromone robotics. *Robotics and Autonomous Systems*, 44:229–240, 2003. ISSN 09218890. doi: 10.1016/S0921-8890(03)00073-3.

Frederick Ducatelle, Gianni A. Di Caro, Carlo Pinciroli, Francesco Mondada, and Luca Gambardella. Communication assisted navigation in robotic swarms: Self-organization and cooperation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4981–4988, 2011a. ISBN 9781612844541. doi: 10.1109/IROS.2011.6048110.

Frederick Ducatelle, Gianni A Di Caro, and Luca M Gambardella. Cooperative Self-Organization in a Heterogeneous Swarm Robotic System, 2010. ISSN 1450300723. URL `http://dl.acm.org/citation.cfm?id=1830501`.

Frederick Ducatelle, Gianni a. Di Caro, Carlo Pinciroli, and Luca M. Gambardella. Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5:73–96, 2011b. ISSN 19353812. doi: 10.1007/s11721-011-0053-0.

F L W Ratnieks and C Anderson. Task partitioning in insect societies. *Insectes Sociaux*, 46(2): 95–108, 1999.

A. G. Hart and F. L. W. Ratnieks. Task partitioning, division of labour and nest compartmentalisation collectively isolate hazardous waste in the leafcutting ant Atta cephalotes. *Behavioral Ecology and Sociobiology*, 49(5):387–392, 2001. ISSN 03405443. doi: 10.1007/s002650000312.

B. R. Johnson. Task partitioning in honey bees: the roles of signals and cues in group-level coordination of action. *Behavioral Ecology*, 21(6):1373–1379, nov 2010. ISSN 1045-2249. doi: 10.1093/beheco/arq138.

Thomas Schmickl and Istvan Karsai. Sting, Carry and Stock: How Corpse Availability Can Regulate De-Centralized Task Allocation in a Ponerine Ant Colony. *PloS one*, 9(12):e114611, jan 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0114611.

Dani Goldberg and Maja J Matarie. Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks. *Robot teams: from diversity to polymorphism*, pages 1–24, 2001.

Giovanni Pini, Arne Brutschy, Mauro Birattari, and Marco Dorigo. Task partitioning in swarms of robots: Reducing performance losses due to interference at shared resources. *Lecture Notes in Electrical Engineering*, 85 LNEE:217–228, 2011. ISSN 18761100. doi: 10.1007/978-3-642-19730-7_15.

Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, 28:101–125, 2014. ISSN 13872532. doi: 10.1007/s10458-012-9212-y.

Matthias Wittlinger and Harald Wolf. The Ant Odometer : Stepping on Stilts and Stumps. *Science*, 312(5782):1965–1968, 2006.

Eliseo Ferrante, Ali Emre Turgut, Edgar Dué, and Marco Dorigo. Evolution of Self-Organized Task Specialization in Robot Swarms. *PLoS Computational Biology*, 11(8):1–21, 2015. doi: 10.1371/journal.pcbi.1004273.

Carlo Pinciroli, Vito Trianni, O'Grady, and et al. ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. *IEEE International Conference on Intelligent Robots and Systems*, pages 5027–5034, 2011.

M Dorigo, D Floreano, L M Gambardella, and et al. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IRIDIA, Brussels, Belgium, Tech. Rep*, 20(4):11–14, 2011.

James Hilder, Rebecca Naylor, Artjoms Rizihs, Daniel Franks, and Jon Timmis. The Pi Swarm : A Low-Cost Platform for Swarm Robotics Research and Education. pages 151–162, 2014. doi: 10.1007/978-3-319-10401-0{_}14.

S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Martín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. ISSN 0031-3203. doi: 10.1016/j.patcog.2014.01.005. URL `http://dx.doi.org/10.1016/j.patcog.2014.01.005`.

Kieran Alden, Mark Read, Paul S Andrews, Jon Timmis, and Mark Coles. Applying spartan to Understand Parameter Uncertainty in Simulations. *The R Journal*, 6(2):1–18, 2014.

Tucker Balch. *Behavioral diversity in learning robot teams*. PhD thesis, 1998. URL `http://www.cs.cmu.edu/{~}trb/papers/thesis/chapter1.ps.Z{%}5Cnhttp://smartech.gatech.edu/handle/1853/6637`.

Kieran Alden, Jason Cosgrove, Mark Coles, and Jon Timmis. Using emulation to engineer and understand simulations of biological systems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2018.

Harold D Delaney and András Vargha. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.

Geoffrey Neumann, Mark Harman, and Simon Poulding. Transformed Vargha-Delaney Effect Size. *International Symposium on Search Based Software Engineering*, 1:318–324, 2015. doi: 10.1007/978-3-319-22183-0.

R. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, 4:264–271, 1987. doi: 10.1109/ROBOT.1987.1088037. URL `http://ieeexplore.ieee.org/document/1088037/`.

Ling Li, Alcherio Martinoli, and Yaser S Abu-mostafa. Diversity and Specialization in. *Proc. of the Second Int. Workshop on Mathematics and Algorithms of Social Insects.*, pages 91–98, 2003.

Tucker Balch. Communication, diversity and learning: Cornerstones of swarm behavior. *International Workshop on Swarm Robotics*, pages 21–30, 2005. ISSN 03029743. doi: 10.1007/b105069.

Josh C Bongard. The Legion System: A Novel Approach to Evolving Heterogeneity for Collective Problem Solving. *European Conference on Genetic Programming*, pages 16–28, 2000.

Alan C. Schultz Mitchell A. Potter, Lisa A. Meeden. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. *International joint conference on artificial intelligence*, 17(1):1337–1343, 2001.

Vito Trianni and Stefano Nolfi. Engineering the evolution of self-organizing behaviors in swarm robotics: a case study. *Artificial life*, 17(3):183–202, 2011. ISSN 1064-5462. doi: 10.1162/artl{_}a{_}00031.

Elio Tuci. Evolutionary Swarm Robotics : Genetic Diversity , Task-Allocation and Task-Switching. pages 98–109, 2014.

Emma Hart, Andreas SW Steyven, and Ben Paechter. Evolution of a functionally diverse swarm via a novel decentralised quality-diversity algorithm. *arXiv preprint arXiv:1804.07655*, 2018.

James Hilder, Alexander Horsfield, Alan G Millard, and Jon Timmis. The Psi Swarm : A Low-Cost Robotics Platform and Its Use in an Education Setting. In *Conference Towards Autonomous Robotic Systems*, pages 158–164. Springer International Publishing, 2016. ISBN 9783319403793. doi: 10.1007/978-3-319-40379-3.

M D Mckay, R J Beckman, and W J Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code. *Technometrics*, 21:239–245, 1998.

A. Saltelli and R. Bollardo. An alternative way to compute Fourier Amplitude Sensitivity Test (FAST). *Comput. Stat. Data Anal.*, 26(4):445–460, 1998.

Michael G B Blum, Oscar E Gaggiotti, and Olivier Franc. Approximate Bayesian Computation (ABC) in practice. *Trends in Ecology and Evolution*, 25(7):410–418, 2010. doi: 10.1016/j.tree.2010.04.001.

Kalyanmoy Deb, Associate Member, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm :. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Ana Duarte, Franz J Weissing, Ido Pen, and Laurent Keller. An evolutionary perspective on self-organized division of labor in social insects. *Annual Review of Ecology, Evolution, and Systematics*, 42:91–110, 2011.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

pages = 91–98 title = Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots year = 2005 Nembrini, Julien.