Cockroach Inspired Shelter Seeking for Flying Swarms of Robots

Hamoon Shahbazi
2015

Master of Science in Engineering Technology
Space Engineering

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Cockroach Inspired Shelter Seeking for Flying Swarms of Robots

Author: Hamoon Shahbazi

Supervisor: Dr. Jan Carlo Barca

A thesis submitted in fulfilment of the requirements for the degree of Master in Space Engineering at the Luleå University of Technology Department of Computer Science, Electrical and Space Engineering

August 2015
“As I have said so many times, God doesn’t play dice with the world.”

Albert Einstein
Cockroach Inspired Shelter Seeking for Flying Swarms of Robots

by Hamoon Shahbazi

In computer science, the study of mimicking nature has given rise to Swarm Intelligence, a distributed system of autonomous agents interacting with each other to collectively perform intelligent tasks. This chapter investigates how groups of holonomic flying robots such as quad copters can seek shelter autonomously when encountering bad weather. In this context three alternative autonomous shelter seeking techniques that address the unsolved plateau-problem had to be implemented. The methods were inspired by cockroaches and hunting strategies observed in apex predators. Previous studies on cockroaches have provided facts about their behaviour and resulted in algorithms that can be used for robotic systems. This research builds on these previous studies by formulating three alternative techniques and carrying out a comprehensive analysis of their performance. Simulation results confirm a scalable system where swarms of flying robots successfully find shelters in 3-D environments.
Acknowledgements

This has been a very exciting journey of my life, moving 15 282 km away from the place I call home. Not only did I put my research skills to the test, but I also made a lot of new friends and connections in the academic world.

First of all I want to thank my friends and colleagues at Monash University and especially my mentor and director of the swarm robotics lab, Dr. Jan Carlo Barca. I’ve never had a mentor that has been so supporting and passionate about his work. It was not an easy project, but with his expertise in swarm robotics I never felt worried to make it. I could not have done all this work without you, Jan.

Thanks to Carlo Pinciromli at IRIDIA of the Université libre de Bruxelles, in Brussels for providing me with the latest version version of the ARGoS simulator and for helping me understand the software.

I would also like to thank my beloved family for making me look ahead to a bright future. My mother, father and sister are always supporting me throughout every decision I make in life. I am as much proud of you as you are of me. To my beautiful girlfriend Amanda, who I also consider part of my family, I want to say thanks for putting up with me moving to the other side of the earth.

I may as well thank my cat, Frank Zappa, for not completely destroying the apartment while I was gone and for keeping in touch via video calls on skype (although when I came back, you seemed to have forgotten me. Why?).

YouTuber Felix Kjellberg, a.k.a ”Pewdiepie”, also deserves thanks for his videos that always kept my spirit up.

Final thanks goes to ÅF (previously knowns as ”Ångpanneföreningen” or ”The Steam Generator Association”) for their financial support in granting me a scholarship.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Algorithms</td>
<td>vii</td>
</tr>
<tr>
<td>Definition of Terms</td>
<td>viii</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>2 Literature Review</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.1 Shelter Seeking and Path Planning</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Previous studies done on cockroaches</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2 Roach Infestation Optimization (RIO)</td>
<td>6</td>
</tr>
<tr>
<td>2.1.3 Randomized Algorithm Mimicking Biased Lone Exploration in Roaches (RAMBLER)</td>
<td>7</td>
</tr>
<tr>
<td>2.1.4 Lévy Flight</td>
<td>7</td>
</tr>
<tr>
<td>2.1.5 Plateau Problem</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Connectivity</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Scalability</td>
<td>13</td>
</tr>
<tr>
<td>2.4 Limitations of Current Techniques</td>
<td>14</td>
</tr>
<tr>
<td>2.5 Challenges for this Research</td>
<td>15</td>
</tr>
<tr>
<td><strong>3 Methodology</strong></td>
<td>16</td>
</tr>
<tr>
<td>3.1 Construct a Conceptual Framework</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Develop a System Architecture</td>
<td>18</td>
</tr>
<tr>
<td>3.3 Analyse and Design the System</td>
<td>19</td>
</tr>
<tr>
<td>3.4 Build the (Prototype) System</td>
<td>19</td>
</tr>
<tr>
<td>3.5 Observe and Evaluate the System</td>
<td>20</td>
</tr>
<tr>
<td>3.6 Research Questions</td>
<td>21</td>
</tr>
<tr>
<td><strong>4 Proposed Technique</strong></td>
<td>22</td>
</tr>
</tbody>
</table>
## Contents

4.1 System Architecture .................................................. 23  
4.2 Standard RIO ............................................................. 24  
4.3 Modified RIO .............................................................. 25  
  4.3.1 Shelter Detection .................................................. 26  
  4.3.2 Sheltered Friends Pull In Neighbours .......................... 27  
  4.3.3 Blind Search With Random Walks ............................... 28  
  4.3.4 Lonely Robots Leave Shelters .................................. 30  
4.4 Gaussian RIO .............................................................. 31  

5 Experiment Design and Results ........................................... 32  
  5.1 Experiment Design Overview ........................................ 32  
  5.1.1 ARGoS Simulator .................................................. 33  
  5.2 Time Efficiency ....................................................... 34  
  5.3 Shelter Connectivity .................................................. 38  
  5.4 Noise Tolerance ....................................................... 40  
    5.4.1 Data Evaluation ................................................ 43  

6 Conclusions and Further Research ..................................... 44  
  6.1 Conclusions ............................................................ 44  
  6.2 Limitations of this Research ....................................... 45  
  6.3 Future Work ............................................................ 45  

A Test 2 - Shelter Configurations ........................................ 47  

Bibliography .................................................................. 52
List of Figures

2.1 Probability Density Functions ........................................ 8
2.2 Lévy vs Brownian Motion ............................................. 8
2.3 Overshooting Lévy Flight ............................................... 9
2.4 Vortex motion ............................................................ 10
2.5 Plateau Problem ......................................................... 11
2.6 Premature Convergence .................................................. 12
2.7 Strong and Weak Connectivity ......................................... 13
2.8 Scalability ................................................................. 14

3.1 Methodology ............................................................... 16

4.1 System Flowchart ......................................................... 23
4.2 Sheltered Friend Behaviour ............................................. 28
4.3 Communication Node Length .......................................... 28
4.4 Lévy and Gaussian Simulation ......................................... 30

5.1 Robots in ARGoS ........................................................... 34
5.2 Message Protocol for MODRIO ......................................... 34
5.3 Test 1 Results For ModRIO ............................................ 36
5.4 Test 1 Results For GaussRIO .......................................... 36
5.5 Test 1 Results For StdRIO ............................................... 37
5.6 Test 1 Results Comparison ............................................. 37
5.7 Test 2 Results Comparison ............................................. 39
5.8 Test 3 Raw Results Comparison ...................................... 41
5.9 Test 3 Results For ModRIO ............................................ 41
5.10 Test 3 Results For GaussRIO ......................................... 42
5.11 Test 3 Results For stdRIO ............................................. 42
5.12 Test 3 Fitted Results Comparison .................................... 43
List of Algorithms

1. Roach Infestation Optimization (RIO) ............................................. 25
2. Modified Roach Infestation Optimization (MODRIO) ...................... 26
Definition of Terms

Centralized system
Is a master-slave based system where all information and commands are passed through the central unit (master). There is no sporadic sharing or decision-making amongst the local nodes without the central power being involved. (CFFN, 2009)

Decentralized system
As in a centralized system, there is an individual decision-making authority that guides the system components. Although, everything is divided in hierarchical order. Thus the central core controls the middle tiers who in turn control the local nodes. The power is decentralized, but the central unit still has control over the entire system. (CFFN, 2009)
**Distributed system**
An assemblage of nodes or computers with equal authority that interact and share their computational resources and activities. From a user perspective one would see it as a single system because the nodes together act as if they were one large computer (Computer Hope, n.d.). The architecture of a distributed system is such that the nodes are autonomous and could in principal work independently. (CFFN, 2009)

![Distributed system diagram](image)

**Heterogeneous swarms of robots**
Is a swarm of unique robots with different capabilities and divergent roles. (C.C Lin 2013)

**Homogenous swarms of robots**
A unity of robots all having identical structure and capability. (C.C Lin 2013)

**Methodology**
Is a structure of methods or rules that can guide the procedure of the work. (Merriam Webster 2014)

**Quad copter**
Is a rotorcraft with four propelled rotors that are mounted symmetrically on a cross frame. By controlling the rotation of the rotors, one can maneuver the rotorcraft. (Hoffmann 2004)

**Swarm Intelligence**
Swarm Intelligence (SI) is a decentralized or a distributed system where multiple units using very simple components can make quick and smart decisions without the help of an external controller. Although the units are equally "smart", the overall intelligence of the group is higher than the intelligence of the single unit. (Gerardo Beni 2005)
Unmanned Aerial System

Nowadays industry has adopted the term Unmanned Aerial systems (UAS) rather than using the previously preferred term unmanned aerial vehicle (UAV), which is an aircraft that does not carry any pilot, but is either autonomous, monitored, or controlled by a pilot from ground or from another vehicle. UAS not only refers to the UAV platform itself, but also to the system as a whole, i.e. the ground-based controller and the communication system between them. (Unmanned Aerial Vehicle System Association n.d.)
Dedicated to my nearest and dearest
Chapter 1

Introduction

Robotic systems have received significant attention over the last decades. We are on the verge of entering a new phase where the next generation of robotic technology is integrated in our daily lives. This may be beneficial in many ways. For instance, when humans are not prone to take on a certain task, robots can be used to avoid exposure to danger. In some cases robots replace humans for repetitive tasks or simply because they are more efficient in strength, speed or accuracy (Campo, 2010). Different types of robots (e.g. crawling, climbing or flying robots) engender their own unique benefits. For example, one of the many distinctive advantages of flying robots is their broad view coverage. In fact, these types of robots are used today to help Japanese farmers monitor their crops and spray pesticides (Greiner, 2013). Other UAS protect the wildlife in Africa by tracking poachers. The future of flying robots looks promising. For instance, swarms of tiny flying robots could quickly, safe and cost efficiently sweep and visually inspect infrastructures such as bridges and dams. The possible applications are almost endless, e.g. firefighting, police and border observation, search and rescue, mapping, radiation detection, damage observation and assessment (UAV Design, 2013). Some of these applications are already in the development stage (Liu et al., 2010). David G. Green (2014) mentions a prosperous future for swarm robotics with the evolution of nanotechnology. He mentions that small nano-bots can extract contaminants from different mixtures in food or even in our bodies (tracking and removing viruses). As the technology in this particular field will continue to evolve, surely the area of application can be extended even further. Perhaps even to planetary exploration, where flying robots can replace ground moving rovers or even satellites in search for life or close mapping on other earth-like planets (Pahlsson et al., 2011). Before we can send robot swarms to other planets, we need to make sure they can survive on their own, i.e. make them completely autonomous. Robots used for outdoor missions are exposed to a world of harsh environments. Shelter seeking is for that reason very important if the robots
shall remain independent. It is a self-defensive mechanism that may protect them in situations where there is a risk for mission, or even system failure. Shelters are not always easy to reach, and in some cases the capacity of a shelter is low, forcing the swarm formation to morph, reducing the distance between each robot. For UAVs the minimum distance to objects is crucial to avoid inter-robot collisions and collisions with environmental objects. On top of that, flight instability may occur as a consequence of turbulent air, from e.g. the down-wash of a neighbouring rotor craft.

In cases where the shelter size is not sufficient to hold the entire swarm it may have to be split into multiple groups. A maximum distance has to be defined in order to prevent the robots from losing contact. Here the term connectivity becomes a key component in a dynamic swarm. A connected swarm ensures the continuous flow of information throughout the whole swarm. It may be interpreted as a condition on how scattered a swarm can become without breaking up into sub-swarms. This is important for each individual within the group to keep improving its own current position by using its neighbours as references.

We can now see that shelter seeking is a complex behaviour to implement in a swarm of robots. The lack of research in the field extends the difficulty even further. The problem is solvable though. In fact it has already been solved, by nature itself. For example, a cockroach has the ability to effectively seek shelter with the help of its friends and its love for darkness. Dark areas are usually signs of potentially good shelters against bad weather, e.g. rain and harsh winds, which roaches dislike and try to avoid (Ganihar et al., 1994). We can relate this to flying robots that may experience flight instability or electrical malfunctions due to turbulence, rain, lightning and other harsh weather conditions. Because biological systems are so complex, biomimetics is applicable for a vast variety of fields. Over extremely long time and through natural selection, nature has evolved and adapted to solve engineering problems. Seeking inspiration from nature is a great place to start, because in most cases life has tackled similar problems which we are facing today. With traditional engineering methods it may become extremely difficult to design a new system because all possible scenarios have to be taken into consideration. In space engineering there is no room for mistakes. In 1961, during the Americas space race with the Soviet Union, an astronaut named Alan B. Shepard was sitting in the Mercury capsule in hope of becoming America’s first man in space (Green, 2014). There had previously been little success and the space programme was therefore hanging by a thread. Because of the complexity of the thousand parts of the rocket system, the technicians had to check and recheck everything to make sure there was not going to be any failure. Due to all the system analysing the coffee that Shepard had been drinking four hours earlier, made his bladder full. He had to relieve himself in the suit, which caused some problems with the electrical equipment that was attached to
his suit for medical purposes. The rocket was launched and in the end everything went fine. It still needs to be emphasized that it could have ended badly had the equipment short circuited and Shepard’s suit caught fire. The engineers were positive that they had thought of everything, but they had not. The traditional engineering technique almost failed because of an unexpected changed scenario. Nature inspired techniques are great because nature has the ability to adapt to changing circumstances. It is therefore both beneficial and convenient to take advantage of living organisms to help solve our own challenges, or at least guide us in the right direction.

Studying insect swarm behaviour is intriguing and has been helpful to find simple solutions to complicated problems before. As Deborah Gordon, a Stanford Professor, said in a CBS news segment in 2011, regarding the study of ants: "Ants are not smart. But colonies are smart..." Simple individual rules are in the collective group amplified, thus a higher intelligence is reached which is otherwise unknown to the single individual (CBS News, 2011; Collins English Dictionary n.d.). A swarm robot does not need to be too complex and mass production of the same robot can be made. Hence, by exploiting the properties of swarm systems one may use fewer resources. Other benefits may involve using multiple agents to carry payloads exceeding the capacity of one individual agent (Kumar & Michael, 2012). One of the most significant benefits of being able to use a large amount of robots is faster exploration performances. A big swarm can cover a greater search area than a small one, although this may be at the cost of group coordination (Bjerknes & Winfield, 2013).

One major challenge for autonomous flying swarms of robots is being able to navigate, detect and avoid any obstacle in the shelter finding trajectory. Outdoor environments are difficult to operate in and affect the robots with limited sight, wind and direct sunlight. If the robots are using navigation systems such as GPS, there is a great chance that they may be shadowed. All of these issues are severe and may in fact lead to a total failure of an individual robot. One may think to add additional on-board sensors as a safety precaution, but that takes its toll on both power consumption and computational resources.

As previously mentioned, the technology of autonomous UAS is constantly evolving, leading to reductions in size, weight and power consumption. While these challenges are obvious and addressed, other issues such as scalability, are easily forgotten. It is a term that refers to the systems limit of attaining coordination amongst the group. It has been found that larger groups need considerably more interactions between the group members to coordinate, than a group with less number of entities (Klavins, 2004). Since interactions require sharing information received from different sensors, this process
exerts a lot of time and resources which in turn affects the coordination of the group. Self-organizing systems involve minimal selection of interactions, hence is not as dependent on the group size. While coordination can and has been proven successful in centralized systems, it is intrinsically weak. For example, a leader receives information from its subordinates, processes the data and makes a decision which is ordered out to the whole team. Since this has a single point of failure, it is also unreliable should e.g. communication with the leader be lost. A distributed system is more redundant in the sense that it does not require any central processor to collect all data or perform iterations. This type of systems is preferable for shelter seeking, as we want a more robust system than that of a centralized architecture.

The importance of shelter seeking, obstacle avoidance, connectivity and scalability is clear. That is why this thesis focuses on finding an effective shelter seeking method that is compatible with obstacle avoidance techniques while keeping the inter-robot connectivity intact, regardless of the amount of robots. The result is a self-governed system of flying robots. The problems we will tackle are nonlinear decision-making processes that may introduce difficulties in the form of confusion and uncertainty of the local best position known to the swarm. A way to counter this complexity is by utilizing the self-organization pattern found in insects, in particular cockroaches, and animal predator societies.

Through interactions with neighbours and without being governed by any leader, this stochastic system results in a unified decision made by a group with limited information (Campo, 2010). The collective decision-making enables effective ways for a swarm of cockroaches or robots to agree upon an appropriate shelter, without an intricate network of sensors.

The reason for choosing roaches over any other insect, besides their great survival abilities, is that they are harmless, easily accessible and cheap. This makes them popular to use by researchers and as a consequence of that, there are a lot of available papers on experiments and analysis on these creatures.

Before setting up requirements for the system and defining appropriate research questions, relevant information about shelter seeking behaviour of cockroaches, avoidance and connectivity techniques has to be gathered. This is described further in the upcoming chapter.
Chapter 2

Literature Review

In this section we analyse relevant published work done in the general area of cockroach behaviour (both individually and in aggregates) and a clearer definition of connectivity and scalability. Additionally, we will take a look at interesting hunting strategies performed by various predators. From unsolved problems and limitations of existing techniques, research challenges are also defined within this chapter.

2.1 Shelter Seeking and Path Planning

2.1.1 Previous studies done on cockroaches

In 2005 Jeanson et al. published their experiment with the cockroach *Blattella germanica*. They investigated how the interaction between larvae, at the individual level, influenced the formation of aggregates. The results showed that the cockroaches seek shelter in dark places and that they much rather form groups than walk around by themselves. Amé et al. (2006) carried out a similar test where they found that the interaction between individual agents result in a social amplification that leads to optimal formation of groups. The larger the population in a shelter, the lower the probability to enter and to leave the shelter. If multiple shelters are present, rather than forming groups by filling up each shelter to its maximum and leaving the surplus individuals in other shelters, they would fill the shelters more evenly. E.g. in an experiment with 50 cockroaches having three shelters with the capacity of holding 40 roaches each, the solution will be 25 in one shelter, 25 in the second shelter and none in the third. Their result support evidence that without elaborate communication and information about their environment, the cockroaches are able to adapt and form groups. Halloy et al. (2007) introduced cockroach-like robots into a group of cockroaches and managed to
somewhat change the behaviour/choice making of the whole swarm. This is in accordance with the fact that the roaches act in a distributed system with no obvious leader to guide them.

There have been tests done where the cockroach’s individual behaviour has been translated to robots such as in and Garnier et al. (2005). Their experiments showed that the collective behaviour of cockroaches could be simulated by robots programmed to follow simple rules, and the result was a close match between the artificial and biological system. Halloy et al. (2007) conducted an experiment by implementing and using the cockroach aggregation behaviour in micro-robots. These robots then managed to change the decision making of a shelter seeking swarm of roaches.

### 2.1.2 Roach Infestation Optimization (RIO)

This made Havens et al. (2008) choose cockroaches as a model for their SI algorithm called **Roach Infestation Optimization** (RIO). It is an algorithm based on the previous stochastic optimization technique from 1995, called **Particle Swarm Optimization** (PSO). That is, a computational model that guides particles in a system towards the best known location continuously found by new particles (Kennedy & Eberhart, 1995). These kind of models are convenient to use when navigating swarms of robots through unknown environments without the aid of an external pilot. RIO follows three basic rules inspired by previous research on cockroaches:

1. Cockroaches **randomly search for the darkest location** in the search space. The level of darkness at a location \( r \in \mathbb{R}^D \) is directly proportional to the value of the fitness function at that location \( F(r) \).

2. Cockroaches **enjoy the company of friends** and socialize with nearby cockroaches with a probability equal to the numerical results of cockroach aggregation behaviour studied by Jeanson et al. (2005).

3. Cockroaches **periodically become hungry and leave** the comfort of darkness or friendship to search for food.

Just like PSO, RIO is a meta-heuristic and does not make any presumptions about the problem that is being optimized (Beheshti & Shamsuddin, 2013). It has a simple structure, with few adjustable parameters which makes it easy to implement and fast in computation. Consequently, RIO inherits the same limitations as its predecessor, namely that it does not guarantee that an optimal solution is ever found. In some complex problems it simply gets stuck in a local optimum due to lack of exploration.
We say that the solution prematurely converges (more on this in the ”Connectivity” section). RIO has somewhat countered this effect by adding a "find food" behaviour. This means that even if a particle is stuck in an optimum, it will eventually get hungry and leave the place to find food. When the particle is no longer hungry it will once again search for an optimum and hopefully find the true global best. Although there are only a few parameters involved, it may still be necessary to tune these to achieve fast and correct convergence for a specific search space. This technique has not been tested on real robots. Although PSO has, there is no record of it being used in a 3-dimensional scenario.

2.1.3 Randomized Algorithm Mimicking Biased Lone Exploration in Roaches (RAMBLER)

Dalterioro et al. (2013) presented an experiment done with the cockroach Blaberus discoidalis, where they chose to investigate what Jeanson et al. did not consider. They wanted to know if there was any shelter-seeking bias model (i.e. if they somehow adjusted their path upon visually observing a shelter) that could fit. The resulting model became the Randomized Algorithm Mimicking Biased Lone Exploration in Roaches (RAMBLER), a more improved randomized walk model than previously made. The main concept of this search is for a robot to walk along walls until it can visualize a shelter with e.g. camera sensors. The technique is not made for three dimensions as ground moving robots with antennas were used in these experiments. The science behind the algorithm only considers a lone cockroach’s navigation in an unknown environment. This precludes any swarm behaviour.

2.1.4 Lévy Flight

The random motion of foraging animals is sometimes said to be similar to a particle moving in a gas. Other theories suggest that the motion has a variety of small and great trajectories or flights.

Lévy flight, just like the swarm optimization algorithms, has random walk behaviour. In contrast to Gaussian or Brownian diffusion (gas particles), a Lévy distribution has a probability to make the random walking unit jump a step length \( l_j \) drawn from a probability density function which follows a power law in its tail. This gives it a fat-tailed distribution, or a large skewness compared to a normal distribution and the density function decays at large \( l \) (see Figure 2.1).
Chapter 2. Literature Review

Figure 2.1: The left side plots show the Lévy probability density function with different scale parameters. Observe that the long fat exponentially decaying tail is only on one side of the function, causing a skewness in the graph. This can be compared with the standard normal distribution (Gaussian distribution) plotted on the right side. The Lévy distribution has the majority of its probability mass under higher values. Consequently there is a chance for a super-long step to occur (this is the so called Levy flight) as we expect a sample from this distribution to return a relatively high value.

This leads to super-diffusive search patterns where a particle is more likely to explore a larger area of the search space (Figure 2.2).

Figure 2.2: A particle moves randomly in a 2-dimensional search space. In a) the particle draws its movement from the Lévy distribution and therefore sometimes experiences a large jump to a new area. In b) the particle exhibits a Brownian movement drawn from a normal distribution. In this case the particle covers less ground as it usually has smaller step sizes.

Viswanathan et al. (1999) state the hypothesis that since Lévy flights optimize random searches, biological organisms must therefore have evolved to exploit Lévy flights. While Lévy flight has yet to be proved to actually be used by living creatures, extensive studies on animals such as, eagles, spider monkeys, sharks, honey bees, wolfs etc. show tendencies towards Levy flight or at least to a process with similar behaviour. Another
perfect example that describes the process of Lévy flight is the fast spreading of diseases due to high air traffic connectivity, where viruses and bacteria literally make a "flight" (large step size) to a remote location. Lévy flight is still an idealization of actual natural foraging. This is because the jump steps are straight vector-like, while in real scenarios a curvature of some degree should exist.

As with search algorithms using normal distribution, Lévy flights inherit the property of occasionally overshooting a target. In fact, in some situations the large jumps can be more troublesome than beneficial. An example of this is when a Lévy flight is made so that the target is lost because the distance is too great to detect with any sensor (see Figure 2.3). Because of this the scale parameter of the probability density function needs to be tuned to fit the search space.

![Figure 2.3: The target (star) is overshot by a distance D. In some unfortunate cases, the distance D may become so large that the target is left unexplored.](image)

In an article by Lin et al. (2012) it is presented how a chaotic sequence and a Lévy random process combined and successfully merged into a bat swarm search algorithm, can be used as an effective optimization model.

### 2.1.5 Plateau Problem

The fitness function incorporated in meta-heuristics such as RIO and PSO is used to evaluate the fitness of each particle, i.e. how well it is doing in its translation from previous position to the current one. For example, PSO can be used to minimize the commonly used test function; the Rosenbrock function \( f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \). As a particle randomly walk in the function domain it constantly checks if the fitness function \( f(x, y) \) in the new position is lower than it was in the previous position. If
true, that position becomes the new best known position. Whenever the particle moves further away from the target it is biased towards the latest best known location and is more likely to be dragged back to that position and try moving in another direction. By continuing to examine its own fitness, the particle moves in a vortex like manner (Figure 2.4) towards the lowest point in function $f(x, y)$, which is at the point $f(1, 1)$.

![Figure 2.4: One particle is moving about the search space. Sometimes the particle overshoots the target but due to the fact that it “feels” less fit, it is pulled back towards its personal best (and also towards its neighbours best position). The result is a spiral around the target, which in this figure is at $x = 0$ and $y = 0$.](image)

The solvable problems for this type are restricted to those where it is possible to compare two different points in the search space. We will henceforth refer to this as gradient problems, since wherever the particle traverses there is always a gradient or a force pulling it towards a better known area. A search space where there is no such gradient is from here on described as plateau problems. Every route the particle takes will yield the same fitness, thus the particle has no idea if it did good or bad. Obviously the particle cannot share its experience with the neighbours around it since they are on the same plateau level. Figure 2.5 below illustrates the gradient and plateau problem.
I know there’s a better place this way

I’m at global optimum

Figure 2.5: The particles (dots) move around freely in the search space. The goal is to reach as low as possible, hence if a particle moves in the wrong direction it notices this and can correct its trajectory. On the right side the particles are trapped, they cannot decide which way is the best. A particle cannot even rely on its neighbours since they are in the same situation.

2.2 Connectivity

The most important feature of functional self-governed multi-robot system is perhaps the ability to communicate between each robot. Gazi and Passino (2011) present two different neighbourhood topologies; local and global neighbourhoods. In a local neighbourhood every particle has its own neighbourhood best position, since a particle is only connected to a subset of neighbours in the swarm. A global neighbourhood topology on the other hand shares only one neighbourhood best position, a global best position. This is because every particle is in direct contact with all other individuals in the swarm. It has been stated that a search algorithm using PSO converges faster if the neighbourhood topology is that of a global type, rather than that of a local one. However, with a global neighbourhood approach, the solution is somewhat susceptible to converge in a local optimum instead of a global. Since this point would not represent the actual best point in a given search space, we say that the solution has prematurely converged (Figure 2.6).

It is possible to use both topologies to get their respective benefits. We call that using a dynamic neighbourhood, which means that the topology changes from one to another with time or as the optimization process evolves. For instance, if we want to reduce
Chapter 2. Literature Review

Figure 2.6: One of the particles in the swarm was exploring and heading towards the best solution in the search space when a connected neighbour found an optima. Because the particle’s current position is less fit than its neighbour’s, an attractive force pulls this particle towards its neighbour. Not knowing that this is only a local optimum, the swarm has now prematurely converged.

the risk for premature convergence, we can simply start off with a local neighbourhood topology. Since the particles are not directly connected to every other swarm particle, they are less prone to be pulled in by a neighbour that has found a local best position. This results in a more exploratory start of the search, enabling optima that are difficult to detect in some search spaces, to be reached. The number of neighbours can then gradually be increased until we have a fully connected topology and one global best solution (or best solution found by the swarm).

One can measure the connectivity in a swarm by introducing the concepts of strong connection and weak connection. To achieve a strong connection within a swarm it is necessary for each individual to get information from every other friend in the swarm. This does not necessarily mean that only a global neighbourhood can achieve a strong connection. A formation which allows a node to be indirectly connected to other nodes (via in-between neighbours) is also considered to be strong.

Without a strong connectivity there is a risk that the swarm cannot improve its estimates because some particles do not have access to the knowledge of other particles. Thus we can say that a swarm must be strongly connected to stay together, otherwise it will break down into sub-swarms (which is desirable in some cases). Figure 2.7 below shows examples of both weak and strong connectivities.
2.3 Scalability

The ability to work at a variety of scales, i.e. different number of robots, degree of co-operativity, loads, etc. without losing an excessive amount of reliability or performance, is considerably desired in a system of swarm robots. Scalability is the general ability of the system to maintain a specific relationship between such scaling parameters and the collective performance (Kernbach, 2013). Roughly there are four different types of scalability, namely:

1. Unscalable
2. Scalable
3. Super-scalable
4. Hyper-scalable
In an unscalable system the collective communication and cooperation between the robots is weak, resulting in system failure or performances with significantly weaker efficiency as the swarm grows in number. Super-scalable systems are ideal to produce. The characteristic of such a type is the almost constant performance regardless of the quantity of robots. Normally one would notice a marginally reduced performance with increasing number of units, a so called scalable system. Hyper-scalability is defined by the properties of enhanced performance while additional robots are introduced to the system. The scalability types are illustrated in Figure 2.8.

It is quite rare to find swarm systems or any other collective systems that are super- or hyper-scalable. Those systems, sooner or later, exhibit a bottleneck effect with higher scaling parameters. Decent engineering can provide for high scalability, by keeping the system simple and efficient. Often one has to sacrifice some system functionalities, or hardware, to achieve this.

### 2.4 Limitations of Current Techniques

When summing up the weaknesses one can conclude that none of the reviewed shelter seeking techniques have been tested on quad copters and some of them restricted to two dimensions and single robot searches, instead of swarms. Meta-heuristics such as RIO are vulnerable to the plateau problem, while RAMBLER is adapted to navigate through them.
RAMBLER has been tested on real robots but is only optimized for a single individual, while RIO considers an entire swarm.

### 2.5 Challenges for this Research

The research comes with several challenges. To address these issues a number of requirements were devised. From many requirements two main challenges were defined to support the objectives of this research.

1. The system shall be able to navigate autonomously through an unknown 3-dimensional environment to find the nearest best known location, even in flat regions of the search space.

2. The system should be scalable with up to 25 robots.

None of the techniques previously described can fulfill these requirements by themselves. Even so, these techniques have (with modification or combination) the potential of fulfilling the requirements. Since there are alternate proposed solutions to this system, we need to evaluate each of them to gain the knowledge of how well each solution works, what modifications need to be made and which ones may be combined. The most propitious solution will be the one used in the system design. See chapter 3.3 Analyse and Design the System.

The remaining parts of this thesis are outlined as follows; first, the methodology used to structure this research is presented. Next, is a presentation of the shelter mechanism together with the architecture of three different shelter seeking methods. This is then followed by the experiment design and result section which describes the experiment and simulation tools used. As a final point, a conclusion is given together with suggestions for future work.
Chapter 3

Methodology

The System development research methodology by Nunamaker and Chen (1991) was accepted into the thesis work. It was determined to be suitable for this type of research as it helps structure the system development progress and also provides an iterative path for the system to evolve.

Figure 3.1 presents the stage progression in this particular methodology. It also shows the option to iteratively refine a stage that has already been processed. This is a eminent way to improve the system after gaining more perspicacity in the subject area.

![Figure 3.1: The five main steps in the System Development Research Methodology. Courtesy of Nunamaker and Cheng (1991).](image)

The stages are described further in the sections below.

3.1 Construct a Conceptual Framework

According to the System Development Research Methodology this step involves studying relevant disciplines to understand and identify any flaws in the existing technique. Once this has been comprehended one can get insight to research objectives, new approaches and preferably original ideas.
This research began by studying a wide selection of publications in the general field of swarm intelligence and robotics. By declaring pertinent research questions and formulating reasonable functionalities and requirements, the techniques best fitting the system could be narrowed down from countless to just a few.

By then it was clear that the main issue for developing a functional system was: Finding global optimal areas, while avoiding to get stuck in a flat region in any search space. So far we have only been describing RIO and RAMBLER as pathfinder techniques, although many other not biological inspired algorithms have been encountered during the initial stage of the methodology, such as A* and D* (Stentz, 1993).

RAMBLER biases the individual robot towards the unknown target location via wall-following actions. This makes it perfect for exploration-purposes in this thesis. Unfortunately its major drawback is that it lacks social swarm behaviour, hence its efficiency in a network of cooperating robots is revealed.

RIO embraces the swarm behaviour but has its own limitations; premature convergences and of course the plateau problem. This is quite troubling for a swarm of robots placed in an unknown environment. The difficulty of the problem raises even further if the robots are placed in an environment where global positioning is limited. To prepare them for this, we have to make each robot independent of GPS sensors. This creates a dilemma, because in contrast to RAMBLER, a particle’s position is crucial data for this type of search algorithm. It is like being told to go to a point while wearing a blindfold and without even knowing where you are. It seems impossible, but perhaps we are not looking at the problem in the right way. The robots may search blindly in such a search space, but as soon as they encounter a shelter, they can feel it (dark, dry, no winds and hopefully a lot of happy friends present). Ergo we do not need to know where we are, we just need to know what we are looking for. Then connectivity may perhaps be the key to solving the problem. As RAMBLER would use walls to guide its way through the unknown, a RIO-based technique could make use of its friends.

The research challenges were addressed by designing corresponding research objectives to each challenge.

1. The system shall be able to navigate autonomously through an unknown 3-dimensional environment to find the nearest best known location, even in flat regions of the search space.

   • Devise new swarm behaviours that can bypass flat regions in a 3-dimensional search space.
• Modify the swarm exploration or escape local optima to ensure that all optima can be discovered.

2. The system should be scalable with up to 25 robots.

• Create a robot communication network that does not decrease considerably in performance with increased number of robots.

3.2 Develop a System Architecture

Here is where one defines functionalities of the system components and the relationships between them. The solutions should satisfy the first stage of the methodology.

To achieve the research objective for countering the plateau problem, a pulling force from nearby sheltered friends needs to be implemented. "Go to sheltered friend" was identified as a suitable strategy for pulling searching robots over the edge of the plateau (see Figure 2.5).

To complete the research objective regarding exploration the technique needs to have ways to extend their search area or to escape local optima. Three types of strategies were devised:

i. Find food behaviour (originated from RIO).

ii. Leave shelter behaviour.

iii. Lévy flight.

The scalability challenge was addressed by developing:

• A low level network communication protocol used by all agents.

"Go to sheltered friend"-behaviour, leave shelter behaviour and the networking will need robots to be equipped with wireless communication and a range and bearing system.

There are existing techniques where the collision avoidance is already integrated into the path finder. An example of this is the previously mentioned PSO-CRS, where the swarm algorithm was modified. A similar modification of the PSO algorithm was also conducted on flying robots by Liu et al.(2010). Evidently, obstacle avoidance techniques may in fact be incorporated into a final product of our own modification of RIO. For this reason a RIO-based search algorithm sets a good foundation for future use with integrated collision avoidance techniques.
3.3 Analyse and Design the System

In this stage the modelling of alternative solutions takes place. By evolving alternative solutions and trying them on the intended task, an evaluation of the discrete techniques should eventually lead to the most appropriate solution that can be implemented in the work.

Two different strategies of "Go to sheltered friend"-behaviour were devised. One of these was that the position of the identified friend was used and stored as the individuals best known position, while the other one simply flies directly to that position. The two strategies were evaluated during this phase, and it was found that the first strategy was less time efficient. It was also not liable enough when used together with a find food behaviour, due to the fact that a robot could get hungry before reaching the shelter. Furthermore, the technique would require the use of GPS or some knowledge of what the exact position of the shelter is. With a direct flight the robot always reaches the shelter quickly (without the vortex effect, see Figure 2.4) and can then decide whether to leave or not. As a result of this the latter strategy for "Go to sheltered friend"-behaviour was selected.

The three different types of techniques to widen exploration and to escape local optima were difficult to evaluate individually, especially within the time frame of this thesis. The researcher first proposed a solution that each particle should start in a "gas-mode" to explore. But this requires one to have enough robots to cover the whole search area of interest. RIO/PSO is used when you are unable to cover an area of interest completely and this is most realistic in a real world scenario unless one wants to explore small patches. The decision was finally made to use a hybrid search method with Lévy flight and RIO. Yang & Deb (2010) refer to this method as "Eagle strategy" for stochastic optimization. Since it is yet not a fully accepted method, a comparison with the test results of the same system with normal distribution instead of Lévy, and also with the standard RIO (or as close to the original as possible) shall be made.

3.4 Build the (Prototype) System

In the fourth phase the selected solutions from the previous stage should be fused into a prototype system. Through the process of building a prototype system one can ascertain drawbacks and benefits of the current concept, framework and chosen designs. These insights are crucial for remodelling and improving the system. For example, the initial method to use the standard RIO algorithm was impossible to follow since it could not handle the flat regions in our search space. A modification had to be done to improve
the system. This insight was the start point for this researcher’s understanding of the plateau problem, as well as the formation of the main research objectives.

It was discovered that the networking/communication between the swarm robots was a very complex problem when dealing with high scalability. Chain reactions from robots telling nearby friends to follow them to new possible shelters would easily get stuck in a loop, consequently leading to robots following robots that were in a blind search. Empirical results showed that the communication system could not be longer than three robots to be considered scalable.

With the new modified RIO we introduced new tunable parameters, such as the probability to stay in shelter depending on how many friends that are present. Additionally, it was found that the scale parameter of the Lévy probability density function needed to be tuned to fit the volume of the search space. These parameters were tuned by testing the prototype system and evaluating the results in the next stage.

3.5 Observe and Evaluate the System

This is the fifth and final stage of the Systems development research methodology. The system is tested and observed so that it responds to the defined research objectives in the first stage. If it does not meet the necessary requirements, suggestions for new solutions to improve the performance of the system also are made at this stage.

Three separate tests were developed to evaluate the performance of the new technique. These tests would show the time efficiency, swarm connectivity, scalability and noise tolerance of the system. The results were also compared with the other devised techniques (Gaussian and standard RIO).

The proposed technique will be described in chapter 4. In chapter 6 the strength and weaknesses of the new technique are identified along with a discussion on what can be improved in future work.
3.6 Research Questions

To focus the research in a certain direction, a set of research questions were formulated. These questions need to be addressed in order to satisfy the system requirements.

A. Can a swarm of robots, inspired by nature, seek out appropriate shelters without getting stuck in local optima or flat regions in a 3-D search space?

B. Is the system scalable with up to 25 flying robots in a swarm?
Chapter 4

Proposed Technique

Three alternative techniques that can be used to address the shelter seeking problem are described in this section. The hybrid systems, henceforth referred to as *ModRIO*, *GaussRIO* and *StdRIO* allow the robots to follow three basic instincts:

- Random walks
- Hang on to "good friends"
- Leave shelter or feel hunger

The systems have to address the issue of finding a proper shelter and at the same time being able to navigate through flat regions in the search space as well as escaping possible local optima. Each system’s strength shall be measured by comparisons of the two other similar devised techniques.

A flowchart, relevant to all three techniques, showing the core processes of the system is presented in Figure 4.1.
4.1 System Architecture

In an ideal case we would use the RIO algorithm without making any alternations to the algorithm, but due to the plateau problem and the fact that we preferably want to be independent of positioning sensors such as GPS, this is not possible.

The three modified RIO systems devised to fit this research disregard the fitness during their search, as the robots will only experience two discrete types of fitness; fit or not fit. The robot is either inside or outside a shelter, there is no in-between. To determine whether or not a robot finds itself inside a shelter, a darkness threshold, \( dark_{th} \), is defined. Depending on the desired quality of the shelter, the threshold can be adjusted to lower values (darker areas). If the input data from the light sensor is higher than the threshold (row 16, Algorithm 2) each neighbour message (see message protocol in Figure 5.2) in the first bit is evaluated.
Chapter 4. Proposed Technique

If the input data on the other hand is below the selected threshold level (row 26, Algorithm 2), the robots stop and becomes an attracting beacon. The robot then counts how many robots that are currently inside the shelter and calculates a probability to leave the shelter.

The robot chooses to leave or stay after comparing the value with a random number drawn from the uniform distribution. When the robot leaves, it resets to its initial state, travels to a random food location and runs the algorithm all over again to find a shelter.

The maximum number of robots relies on both the desired performance of the swarm and also on the size of the search space. Experiment results that show a trend of significantly fast convergence or hyper scalability can be used to decide the quantity of robots that saturate the selected search volume.

4.2 Standard RIO

Standard RIO, or StdRIO, is designed to mimic the original RIO algorithm. Here an exception is made, where positioning sensors (i.e. GPS) are allowed to be used by the system. The technique uses memory to store the position of the personal and local best known shelter location. The leave shelter behaviour is replaced with a find food behaviour, which sends out hungry robots to search for food in different intervals. Because this technique is specially dependent on memory it breaks for robots that are alone and have no best known position. For these cases it is therefore altered to follow the design of blind random walk that is drawn from the normal distribution, until shelter positions can be established.

The parameters for StdRIO are setup according to the original RIO presented in Algorithm 1 below.
Algorithm 1 Roach Infestation Optimization (RIO)

Input: Fitness Function $F(\vec{x}) \in \mathbb{R}^D$

Parameters:
- $N_A$
- $t_{max}$
- $C_0$, $C_{max}$
- $A_1$, $A_2$, $A_3$

5: $t_{hunger}$

Initialization:
- Set $hunger_i = \text{rand}\{0, t_{hunger}-1\}$
- Set population $\vec{x}_i$ and $\vec{v}_i$, randomly
- Set food location $\vec{b}$ randomly

for $1 \leq t \leq t_{max}$ do

10: $M = [M_{jk}] = \|\|\vec{x}_j - \vec{x}_k\|\|_2$
- $d_g = \text{median}\{M_{jk} : 1 \leq j < k \leq N_A\}$

for $1 \leq i \leq N_A$ do

if $F(\vec{x}_i) \leq F(\vec{p}_i)$ then $\vec{p}_i = \vec{x}_i$

- Compute the neighbours of roach $i$ as
  - $\{j\} = \{k : 1 \leq k \leq N_A, k \neq i, M_{ik} < d_g\}$
  - $N_i = \text{Number of neighbours } |\{j\}|$

for $1 \leq q \leq N_i$ do

if $\text{rand}[0, 1] < A_{min(N,3)}$ then
  - $i_q = \text{argmin}_k\{F(\vec{p}_k)\}, k = \{i, j_q\}$

20: if $hunger_i < t_{hunger}$ then

- $\vec{v}_i = C_0 \vec{v}_i + C_{max} \vec{R}_1(\vec{p}_i - \vec{x}_i) + C_{max} \vec{R}_2(\vec{l}_i - \vec{x}_i)$
- $\vec{x}_i = \vec{x}_i + \vec{v}_i$

else

- $\vec{x}_i = \text{Random food location } \vec{b}$

25: if Hungry then

Increment $hunger_i$ counters

4.3 Modified RIO

To describe the Modified RIO system, also known as ModRIO, we will divide it into four parts. Each part represents the basic instincts of the robots as previously mentioned in the beginning of this head section. Algorithm 2 shows the structure of the system.
Algorithm 2 Modified Roach Infestation Optimization (MODRIO)

Input: Data from light sensors, $I_s$

Parameters:
- $N_A$
- $t_{max}$
- $C_0$, $C_{max}$
- $P_s$

Initialization:
1. Set the population $\vec{x}_i$ and $\vec{v}_i$, randomly.
2. Set the food location $\vec{b}$ randomly.
3. Set all robot messages, $msg_s = false$, $msg_f = false$.
4. Set the darkness threshold, $darkth$, for minimum shelter darkness quality.

for $1 \leq t \leq t_{max}$ do

10. $M = [M_{jk}] = ||\vec{x}_j - \vec{x}_k||^2$
11. $d_g = \text{median}\{M_{jk} \in M : 1 \leq j < k \leq N_A\}$
12. for $1 \leq i \leq N_A$ do
13. Compute the neighbours of roach $i$ as $\{j\} = \{k : 1 \leq k \leq N_A, k \neq i, M_{ik} < d_g\}$
14. if $I_s > darkth$ then $msg_s = false$
15. for $1 \leq q \leq N_i$ do
16. if $msg_{sq} = true$ then
17. $\vec{x}_i = \vec{x}_{jq}$
18. $msg_f = true$
19. else if $msg_{fq} = true$ and $msg_f = false$ then
20. $\vec{x}_i = \vec{x}_{jq}$
21. else
22. $msg_f = false$
23. else
24. $\vec{v}_i = C_0\vec{v}_i + C_{max}\vec{R}(\vec{x}_i + \text{Lévy})$
25. \[msg_s = true\]
26. prob = $P_s \times (N_i \epsilon \text{Shelter} + 1)$
27. if $\text{rand}[0, 1] > \text{prob}$ then
28. $msg_s = false$
29. $msg_f = false$
30. $\vec{x}_i = \text{Random food location } \vec{b}$
31. Relocate eaten food randomly.

4.3.1 Shelter Detection

This technique only requires light and relative positioning sensors mounted on the robots to remain functional. The data from the light sensors $I_s$ varies with the light intensity according to the inverse square law $I(r) = I_{source} / r^2$. Input from these sensors is used to determine if the robot is inside a shelter or not, i.e. if the darkness level is high enough. If a shelter is not detected the robot looks for any neighbouring robot...
that already resides in a shelter (hereafter referred to as shelter robots). This is done by analysing incoming messages from nearby robots via the relative positioning system. When a robot eventually reaches an area that is dark enough it stops and becomes an attracting beacon by transmitting a shelter message. If several shelter robots are discovered the robots chooses to move towards the nearest one. Robots that have no sheltered friends can instead follow neighbours that are in contact with a robot inside a shelter. Those robots transmit a different message (row 21 in algorithm 2).

### 4.3.2 Sheltered Friends Pull In Neighbours

In the case that a sheltered friend is discovered the action is simply to travel to the source of the received message. When using optimization techniques to solve mathematical problems, the algorithms infer that particles ”teleport” or travel with infinite velocity to any position. This is not plausible for real life implementations, thus a speed limit has to be introduced. By setting a constraint on the velocity update equation, it might take several system loops for a robot to reach a spotted friend unlike when performing instantaneous travel. During this time a robot may also lose contact with the target and return to a blind random walk again. The range and bearing system provides enough information for the robots to create a three-dimensional vector pointing to the transmitting robot. The robot could then add thrust to its rotors and make the necessary flight trajectory. Eventually the robot will be pulled into a shelter, start sending out shelter messages and thus becomes a beacon itself for other passing robots (Figure 4.2). The networking can be extended even further by letting other robots know when a sheltered friend has been found, or even when a robot has discovered another robot that has seen a sheltered neighbour, etc. An endless chain reaction can alert the whole swarm. This type of networking may become problematic since it can decrease the exploration of the swarm by frequently pulling individuals towards already discovered shelters. Additionally, in case of lost connections in a distributed system, robots were fooled to think that a good leading robot still existed and started to follow each other as a consequence.

That behaviour was avoided when running the system with a centralized or a decentralized communication network (as presented in Figure 4.3). That is because the hierarchy system does not allow inter slave communication. Thus, the message chain immediately breaks when a master neighbour loses contact with a shelter robot for instance. A node length of maximum two decentralized robots was tested (i.e. one shelter robot pulling an outside friend who in turn attract its own share of blind robots). This solution was still capable of achieving strong connectivity within the swarm without splitting the swarm.
Figure 4.2: A simulation where each robot performs a randomized search for a shelter. Whenever a robot finds itself in a shelter, it immediately becomes a beacon for other connected robots. Robots who are not in direct line of sight are attracted to their neighbours who themselves have discovered a sheltered robot. Thus, the swarm converges into one shelter.

...into several sub swarms, and was for that reason accepted as a solution for ModRIO. For very large search areas the node length needs to be adjusted to accept larger chains.

Figure 4.3: In the decentralized communication model slaves do not influence each other. When a central robot loses contact with its master, its slaves will stop following it. The chain can be made infinitely long, but for each new central level more communication data is required.

4.3.3 Blind Search With Random Walks

In large search areas, it is likely that some robots are unable to sense a shelter or a sheltered friend. Such a robot is completely "blind", and has no idea which way is the most suitable way to go. Here we suggest that the robot takes a random walk either from a Lévy distribution.

When a robot finds itself in a situation where no surrounding friends can help, or if it is all alone, a random walk is performed by setting the velocity vector to...
\[ \vec{v}_i = C_0 \vec{v}_i + C_{\text{max}} \vec{R}(\vec{x}_i + \text{Lévy}) \] (4.1)

\( \vec{v}_i \) and \( \vec{x}_i \) are the current velocity and position of the robot \( i \), \( C_0 \) and \( C_{\text{max}} \) are both weight parameters where \( C_0 \) is the inertia weight that can be seen as a heavy weight slowing down the movement of the current speed. In previous experiments inertia weights of 0.9 to 0.4 have shown to produce good results (Bansal et al., 2011). Standard optimization algorithms have both cognitive and social \( C_{\text{max}} \) parameter as in Algorithm 1. The algorithm normally provides good results with total \( C_{\text{max}} \leq 4 \) (Carlisle & Dozier, 2011).

\( \vec{R} \) and Lévy are vectors of uniform random numbers drawn from the normal and Lévy distribution respectively. To draw a random number from the distribution a method called rejection sampling is used. In short, a uniform random coordinate from a box covering the Lévy probability density function (PDF) is generated. The dimensions of the box can be set to a height that is equal to the amplitude of the chosen PDF so that it encases all possible values. To cover the horizontal size of the Lévy PDF is impossible since its fat tail stretches to infinity. For these scenarios it is sufficient to cut off the tail at any point where the rate of change can be regarded as 0.

If the point happens to belong to the area beneath the Lévy PDF, the corresponding value from the PDF is chosen. In a way this technique can be compared to throwing dart blindfolded at a target being the Lévy PDF

\[ \sqrt{\frac{c}{2\pi}} \times \sqrt{\frac{e^{-c/2(x-\mu)}}{(x-\mu)^{2/3}}} \] (4.2)

where the shift parameter, \( \mu \), is zero in the standard form and \( c \) is the scale parameter.

In Figure 2.1 the different forms of the function with varied scale parameters is displayed.

The scale parameter is much dependent on the size of the search space, or the so called arena size of the robot experiments. It was chosen so that the maximum value drawn from the distribution does not exceed 1/10th of the arena side length or so that

\[ C_{\text{max}} \vec{R}(\text{Lévy}) \leq \vec{R}(\vec{x}_i) \] (4.3)

This way the maximum jump may be up to twice as large as that of Brownian motion component.

The blind fumbling will continue until either a shelter is reached or a sheltered friend is spotted, as described in Algorithm 2. Yang & Deb (2010) called the method of combining
Lévy walk with stochastic optimization techniques ”Eagle Strategy”. The most essential benefits are robots with higher chance to make larger jumps and less probable to wander around the same area (as in Figure 4.4).

Figure 4.4: Two simulations showing a single robot’s path. The left side uses ModRIO, were it is obvious that the Lévy flights contribute to occasional large jumps, thus making the robot cover more ground. On the right side example the GaussRIO has a more dense search and often searches in areas it has already visited before.

4.3.4 Lonely Robots Leave Shelters

When a robot ultimately finds itself inside a shelter it stops. Similar to cockroaches, the next move is for the individual robot to decide whether to leave the shelter or not. The decision is made based upon how many robots (including itself) that are currently confined in the same shelter

\[
prob = P_s \times (N_{i\epsilon{shelter}} + 1)
\]  

(4.4)

\(P\) is the probability for a robot to stay depending on the base probability, \(P_s\), and the number of robots currently inside the same shelter (neighbours, \(N_i\), and itself).

For every friend the robot senses there is a slightly less chance for the robot to leave, because it feels comfortable around its friends. For a robot that is completely alone in a shelter it is more likely to move on to find a shelter with other robots hiding in it. Empirical results showed that a 100% chance to stay for approximately \(1/5^{th}\) of the total swarm population inside one singe shelter was the most time efficient method for the scale of robots used in this research. For a swarm population of up to 25 robots \(P_s = 0.20\), thus a robot will still consider leaving a shelter with up to three other robots currently inside it. Experiments with a large amount of shelters with various sizes may require \(P_s\) to be tuned differently, since swarms have shown to be able to ”sense” different aggregation proportions (Garnier et al., 2005). This loneliness behaviour is comparable
to the "find food" behaviour of the original RIO, where a cockroach over time becomes
hungry to the point that it willingly leaves the comfort of its shelter to look for food.

4.4 Gaussian RIO

Gaussian RIO, or GaussRIO, is identical to ModRIO in every way except that the Lévy
component is removed. As in contrast to ModRIO, random walks are performed by using
the Gaussian (or normal-) distribution. Robots using GaussRIO walk with a Brownian
motion without any random large jumps. The search pattern is therefore more compact,
as in Figure 2.2 b), which might help to keep a higher connectivity within the swarm.

Information about the overall experiment design is provided in chapter 5 below.
Chapter 5

Experiment Design and Results

The first part of this chapter explains the design and layout of the experiments in detail along with simulation tools and parameter values. In the second part the result of these simulation experiments are presented.

5.1 Experiment Design Overview

As described in the methodology section, three tests were performed on each of the three techniques; StdRIO, ModRIO and GaussRIO. Each test was repeated five times to enable us to "see beyond noise". Five repetitions were chosen as this number has been found to offer a good trade-off between accuracy and processing time (Law & McComas, 1990; Hoad et al., 2007).

Prior to each test the robots were uniformly distributed throughout the arena. This means that there is a chance for some robots to start out nearby or even inside a shelter, which may cause lower shelter times. To avoid having tests with different initial advantages between the techniques, they all share the same initial robot distribution.

The number of robots is tested for $1 \leq N_A \leq 25$, because swarms of 26 or more showed tendencies towards a hyper scalable system for the chosen arena size of $10 \times 10 \times 10$ meters. Other values for implementation specific variables are displayed in Table 5.1.

The following three sections will describe the different experiments in detail together with the achieved result.
Table 5.1: Parameters for implementation specific variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Information</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_A$</td>
<td>Number of robots</td>
<td>1 to 25</td>
</tr>
<tr>
<td>$C_0$</td>
<td>Inertia weight</td>
<td>0.7</td>
</tr>
<tr>
<td>$C_{max}$</td>
<td>Roach parameter</td>
<td>1.43</td>
</tr>
<tr>
<td>$P_s$</td>
<td>Base probability</td>
<td>0.20</td>
</tr>
<tr>
<td>$v_{max}$</td>
<td>Velocity constraint</td>
<td>4 (m/s)</td>
</tr>
<tr>
<td>$A_1$</td>
<td>Group parameter</td>
<td>0.49</td>
</tr>
<tr>
<td>$A_2$</td>
<td>...</td>
<td>0.63</td>
</tr>
<tr>
<td>$A_3$</td>
<td>...</td>
<td>0.65</td>
</tr>
<tr>
<td>$t_{hunger}$</td>
<td>Hunger interval</td>
<td>100</td>
</tr>
</tbody>
</table>

5.1.1 ARGoS Simulator

The researcher was offered an exclusive version of the ARGoS 3.0, a multi-robot simulator developed at the Artificial Intelligence research laboratory of the Université Libre de Bruxelles, IRIDIA (Pincioli et al., 2012). It is an open source discrete-time 3D space simulator built on free C++ software libraries. While the main code is in C++, the environment i.e. the arena, physics engines, robots, sensors and other entities are setup using XML configuration files. The robot controller with the robot code was written in Lua scripting language instead of the default C++. Lua has a very simple application programming interface, which made it fast and easy to work with. Regardless if the controller is running simulated or real robot, it is allowed to access the robot on-board devices. The benefit of this is that a compiled code from simulation to real robots is transparent. This is one of the main reasons for choosing ARGoS as simulation platform for this thesis, besides providing high flexibility and computational efficiency making it possible to run on most computers. Another reason is that the researchers were in close contact with the creator of this simulation tool, which helped facilitate the process of getting started with the software.

ARGoS 3.0 allows simulation of three different types of robots; ground moving, climbing and flying (Figure 5.1). One can modify and create own robots, but in this research the Eye-bot model from IRIDIA was chosen.

The range and bearing system allows the robots to perform localized communication. That means that when a robot receives a message it can at the same time distinguish the position of the sender with respect to its own point of view. For this reason, the robots need to be in direct line of sight to communicate. When robots connect, they can exchange data, vertical angles, horizontal angles and distance to the message source (Figure 5.2).
Figure 5.1: The robots that have been modelled inside the ARGoS simulator are based on real robots.

Figure 5.2: The package of information received from each robot. The robots can send a maximum of 10 bytes of data, but do only utilize one of these in the ModRIO and GaussRIO system. In addition to this the robot, via the range and bearing system, also receives the vertical angle, the horizontal angle and the range from itself to the source of the message.

For StdRIO an additional positioning sensor was used for the robot to recognize its exact position.

5.2 Time Efficiency

In the first test three corner shelters were used, similar to the previous experiments done with real cockroaches. Each shelter could house the entire swarm; this was to prevent any robot from being denied entry to an approached shelter. When all robots had been initially placed the experiment commenced, and the time it took for the robot to reach a shelter was recorded (this will be referred to as "shelter time"). The test continued by increasing the number of robots by one, repeating the test, until 25 robots had been initiated. When performing tests with multiple robots they are likely to find shelters at different time. Some robots might even leave their shelters at several occasions by
the time that another robot reaches one. Hence, when all of the robots have visited a shelter at least once, the experiment is stopped and the time is recorded. The shelter time is of great importance for this research since it provides knowledge on how long time beforehand the robots need to predict harsh weather conditions.

For the first test the trend line is represented by a third degree polynomial model. All three techniques indicate a performance growth with increasing number of robots. In minor groups GaussRIO does not show a distinguishing sign of improved performance but instead indicate a marginally slower shelter time for 15 to 16 agents in relation to its initial performance with fewer robots. Nevertheless, together with ModRIO, this technique can be considered the most constant one throughout the experiment, although they both show a slight decline in shelter time at higher populations. StdRIO on the other hand has the slowest shelter time for small groups, with nearly two times higher than that of GaussRIO for a single robot.

From Figure 5.6 in the first experiment we can establish the fact that there is no significant difference in performance between ModRIO and GaussRIO. This could mean that the effect of Lévy flight may be exaggerated or that an even larger search volume is needed to distinguish their dissimilarities. Nonetheless, the models predict an even shorter amount of shelter time as the swarm population increases, which perhaps indicate that the search space is being saturated with the relatively large amount of robots. The saturation is due to the pulling effect of nearby neighbours, eliminating the exploration of each robot and the entire swarm. The fact that StdRIO showed the highest dependency on larger search groups and worst overall results, might be a sign that individual memory is overrated and may therefore not provide an advantage in all problem domains.
Figure 5.3: A third degree polynomial fit to the measured shelter times in the first experiment for the ModRIO method. The graph shows how the swarm becomes slightly more efficient with a larger number of robots.

Figure 5.4: A fitted curve on the GaussRIO technique, showing slower shelter times for up to 16 robots, but more promising results for even larger groups.
Figure 5.5: For stdRIO the trend line is much steeper with the highest values achieved with small groups of robots. The technique makes a remarkable recovery with larger swarms.

Figure 5.6: A comparison of the trend lines of the three techniques. StdRIO is the most dependent on larger quantities of robots as it finally catches up to the other two methods that show a fairly constant efficiency.
5.3 Shelter Connectivity

The second test used a fix amount of robots and instead varies the number of shelters, capacities and their position in the search space. Each shelter can house an amount of robots so that the entire swarm population can be equally divided into the shelters. Thus for $n$ shelters the capacity of each shelter is $1/n$ fractions, e.g. three shelters can each hold a $3^{rd}$ of the entire swarm population, while 6 shelters can house a $6^{th}$. This forced the swarm to split up so that we could monitor the collective behaviour in a weaker state of connectivity. In cases where $1/n$ would imply fractions of robots, the quote was rounded upwards to the nearest integer.

The number of robots was chosen by the maximum number of shelters, with the aim of having one robot in each shelter in the final experiment. Empirical results showed that for a search volume of this size would become saturated with 10 shelters or more. The decision was therefore made to use no more than nine. The shelter distribution is shown in Appendix A. The shelters were chosen to be placed as far from each other as possible to increase the diffusive search patterns of the robots.

Because the robots are much more likely to leave empty shelters, the swarm will have a hard time to converge when shelters are too small. The leave shelter and find food behaviour was therefore stopped after a time corresponding to the highest shelter time of all techniques from the shelter time experiment, i.e. 166.9 seconds. A robot would after this time stay inside shelters it entered without leaving. Roaming robots would, once more, be given an equal amount of time (166.9 seconds) to find shelter before the simulation ended. After a completed simulation it was investigated how many robots failed to hide.

Both ModRIO and GaussRIO show almost identical behaviours in this experiment. The techniques are successful in keeping the majority of robots safe. Yet, an unexpected dip occurs at seven shelters, with capacities of two robots each, causing a robot loss of roughly 40% of the swarm population for both techniques. StdRIO shows the worst result with the lowest shelter population down to 33% of the entire swarm.

In the second experiment with results displayed in Figure 5.7, shelter memory seemed to cause more damage than good. The robots showed an ignorant behaviour where they always preferred the first shelter they visited, regardless how many friends it contained. In most trials this resulted in robots trying to enter a shelter which was already full, instead of finding a new one. When the time for shelter finding was up, the robots were still flying around some shelters waiting for other robots to leave so they could take their positions. This was never the case for ModRIO and GaussRIO, where there is no memory to remember what shelter was previously visited or not. Thus, those two techniques had
a higher success rate with minimal loss of swarm agents. Still a rather interesting result showed that none of the techniques succeeded well on the seven shelter configuration, where ModRIO and GaussRIO both lost about 40% of their robots. Despite that the setup had the same capacity of two robots as the six shelter configuration, which on average sheltered the entire swarm population for both techniques. The only difference is the added shelter in the middle of the arena. There is a possibility that such a shelter disturbs searching robots when it is full. Since robots have a higher chance of discovering such shelters than those that are on the edge of the search volume, it is usually the one that gets filled first. The remaining robots that get dragged towards that shelter realize that it is full only when they reach it. When they leave the area they may at a later time be pulled back in again, this behaviour can repeat itself until it is too late for the robots to find the corner shelters. This theory would also explain why the nine, five and three shelter configuration also showed signs of robot loss for the two techniques.

![Test 2 - Swarm Connectivity](image)

**Figure 5.7:** ModRIO and GaussRIO have a similar pattern and are successful in most cases except for the seven shelter configuration. Yet again stdRIO stands out and displays the worst results for all nine shelter formations.
5.4 Noise Tolerance

The third and final test measured how noise affects the different techniques. The noise was generated from a uniform distribution and was implemented so that whenever a robot experienced noise, it would result in a complete loss of a data package. The arena was set up as in the shelter time experiment and 12 robots were used, which is approximately half of the maximum population of the first experiment. The noise rate started at 0% and was incremented by four percent to acquire the same step size as in the first experiment.

GaussRIO had the highest performance with some trivial oscillations showing a minor overall decrease in performance. ModRIO more than doubled its shelter time between 0 and 10% noise. The performance diminished with increased noise level and the shelter time was at one point four times higher than that of the noise free test. StdRIO was at first undisturbed by the noise but started to grow in an exponential manner for noise levels above 20%. With permanent communication loss this technique more or less quadrupled, while the other two techniques improved. A forth degree polynomial model is used to present the behaviour trends.

The final experiment surprisingly showed that ModRIO technique was the most vulnerable to noise, although StdRIO exponentially grew and took over at the higher noise levels (see Figure 5.12). The lack of Lévy flight might be a fairly reasonable argument to why GaussRIO would seem unaffected by noise while ModRIO is not. It was discovered during the experiments that the noise created confusion where robots ”hesitated” when following their neighbours. This slow decision making process comes from when the robots lose the connection to a shelter friend because of the noise and starts to perform the blind random walk again. During this period ModRIO with its Lévy flight had time to jump outside the communication range, thus permanently losing its shelter friend. GaussRIO on the other hand is more prone to have a more compact search pattern and can therefore resume its path towards its shelter friend when the connection is regained after the noise. Figure 5.12 also shows that the results from ModRIO and GaussRIO improve near a 100% loss of communication. It is possibly the effect of only performing blind random walks without getting pulled or hesitating because of surrounding neighbours.
Figure 5.8: Average data from the three distinct techniques indicating that GaussRIO is considerably faster.

Figure 5.9: With introduced noise to the system ModRIO has a great leap in shelter time. Although having higher variations in the observed data, the technique follows a similar increasing trend as GaussRIO.
Figure 5.10: Eventhough there is an increase in shelter time, the noise seems to leave GaussRIO generally unperturbed.

Figure 5.11: StdRIO seem to be following the same trend as GaussRIO for low level of noises, until the shelter time accelerates and eventually reaches values four times greater than that of a noise free test.
5.4.1 Data Evaluation

Results from the experiments showed a variety shelter time, even by using the mean value of five consecutive trials for each step the values are very much scattered. Consequently it is difficult to observe a trend for such measurements, even though this is to be expected for experiments that are based on randomness. The curve fitting toolbox, `cftool`, in MATLAB was used to create and compare different trend lines. Evaluating goodness of fitted data can be considered a research topic in itself. In some cases the chosen model should have the reasonable statistics to be considered a "good fit". This involves checking statistics such as sum of square error(SSE), R-square and root mean square error(RMSE), etc. In other cases a graphical approach is the best way to go since it presents an easily observed predicted model. The latter is also the most common approach when presenting results. In this thesis however, both statistical and graphical methods were considered to determine the best fit.

In chapter 6 a conclusion is made based on the results. This chapter also includes suggestions for future work.
Chapter 6

Conclusions and Further Research

In the last chapter the meaning of the results achieved in chapter 5 will be formed into conclusions. The conclusions will be made relative to the research questions stated at the beginning of this thesis. Limitations of the new developed technique will be highlighted and suggestions for future studies within this field of work is presented.

6.1 Conclusions

All of the three proposed systems; StdRIO, ModRIO and GaussRIO are capable of finding shelters for the whole swarm under one minute in average, inside a $1000m^3$ arena with three corner shelters. The results for ModRIO and GaussRIO also revealed tendencies of super-scalable systems for up to 25 robots in a swarm.

Additionally, ModRIO and GaussRIO both allow a swarm of robots to seek out shelters without getting trapped inside local optima and flat regions in a three dimensional search space.

Because of the similar performance between ModRIO and GaussRIO perhaps the implementation of Lévy flight may have no particular influence on a system, or conceivably a larger search volume can prove otherwise. The one single difference that could be interpreted is the greater noise sensitivity of ModRIO, while GaussRIO seems rather unaffected to noise.

It was verified that StdRIO had the worst overall results from the experiments, which makes us question the necessity of individual memory.

This research area is yet to be explored and to this authors knowledge there is a lack of studies done on swarm intelligence for flying robots, particularly in involvement with
the plateau or flat-region problem. It can therefore be said that this research has a meaningful scientific contribution and can be considered a great stepping stone for future development of autonomous flying robot swarms.

### 6.2 Limitations of this Research

The devised system had some difficulties when encountering fully occupied shelters that were easily detected. Such shelters had a negative pulling effect on the robots and interfered with their search for less accessible shelters closer to the edge of the search space.

The Lévy component was not as effective as predicted and essentially made the swarm less receptive to noise that ultimately leads to loss of communication.

Although a lot of time was spent on performing the tests and getting data, a lot more information is needed to produce a just model for the performance. The amount of randomness is very high and a sampling size of hundreds if not thousands is needed to have less scattering data and at a high confidence level.

### 6.3 Future Work

A series of extensions to this research can be undertaken to further investigate shelter seeking behaviours for flying robots. One suggestion is to evaluate the functionality of the proposed techniques across varying arena sizes with real life environments, where the population can be scaled up beyond 25 individuals. A different approach to full shelters may have to be developed to counter disrupted search patterns as was discovered in the connectivity experiment.

The technique could also be combined with collision avoidance techniques.

Presenting different qualities of shelters and investigating how this affects the collective decision-making is would also be an interesting avenue for further research as the goal of the swarm is to find the best suited shelter. This does not necessary mean a darker shelter. Escape behaviour of cockroaches studied by Camhi & Tom (1978) supports the theory that cockroaches run away from wind sources, which would explain why a cockroach runs straight into the nozzle of the vacuum cleaner or moves just before being crushed by a shoe. The agricultural experiment station conducted a test where they found that wind is such a strong repellent that cockroaches even prefer to stay in well-lit areas than dark windy ones (Appel et al., 1997). This can be taken into consideration
to better define a good hiding place. One must also take into account what is harmful to the quad copters themselves, e.g. moisture, dust or other aerosols.

On a final note, in this research the individuals were unable to share their previous experiences with neighbours because they require memory to do so. Although the robots were made more selfish, there is no doubt that teamwork through connectivity and networking plays the largest role in a creating an intelligent swarm without specifically using smart individuals.
Appendix A

Test 2 - Shelter Configurations

The different shelter placements are displayed below.
Bibliography


