# Swarm airports

*Optimizing the airport selection in a drone swarm system*

Anders Rønningstad


Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
60 credits

Department for Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

# Swarm airports

*Optimizing the airport selection in a drone swarm system*

Anders Rønningstad

Swarm airports

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

**Abstract**

In this thesis, swarm airports are introduced as a way of obtaining continuous operation in an autonomous drone swarm. An airport acts as a service station where the drones can replenish their energy autonomously. When incorporating several airports in the same system, one can operate a large drone swarm without the need for human interaction. This simplifies the use of drone swarms in applications such as surveillance and delivery.

As there is little or no previous research on swarm airports, there are many issues that could have been addressed. In this thesis, the focus is on how one can optimize the airport selection when a drone requires a battery change. The thesis proposes a possible solution using a method inspired by response threshold, from the field of swarm intelligence. The advantages of using this method lie in the decentralized control, and ability to operate with limited communication.

In order to perform experiments with the proposed method, a simulation tool was created. The simulation program provides a framework and visualization tool in order to better understand the system, as well as making it easier to compare methods with the same configurations. In order to understand how well the proposed method works, it is compared to benchmark methods inspired by mathematical optimization and random decision making.

The main experiments presented in this thesis demonstrate how the different methods perform in terms of active drones in a system consisting of four swarm airports. The results of the swarm method proved adequate when the airports are spread out to several locations, but the method still requires some improvement in order to achieve the same results as the benchmark methods. When changing the configurations such that all airports are at the same location, the method performs well compared to the benchmark methods, indicating that using swarm optimization can be favorable when solving airport selection problems.

# Acknowledgement

Without the support, weekly discussions, and guidance from my supervisors the process of writing this thesis would definently not have been the same. I would therefore express my inmost gratitude to my suporvisors Kyrre Glette at University of Oslo, Aleksander Simonsen at Norwegian Defence Research Establishment, and Hans Jonas Fossum Moen at Norwegian Defence Research Establishment. Thank you for your inpiration and everything you have taught me throughout the whole thesis.

To my fellow students; I thank you for rubber ducking, discussions and the tips you have shared all the way till the end. I would like to thank family and friends for the patience, motivation, and support you have shown.

Last but definitely not least, my significant other Ingvild Dalseng Halleraker. Thank you for your endless support and all the other help you have provided.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Through the years as technology has become more advanced, drones have been used in a wider range of applications. With improved technology the drones can be smarter, which includes stability, improved energy consumption and more data processing onboard. With improved technology drones can also be made faster, better and cheaper, which results in more people wanting and affording them. With the drones getting smarter and more people buying them there is no doubt that the drones will be used in more advanced systems.

A drone is an unmanned aerial vehicle (UAV) which is either autonomous or controlled by someone [11]. The first UAV created was manufactured in 1916 by the Americans Lawrence and Sperry. For this UAV they developed a gyroscope in order to stabilize the aircraft and in order to have altitude control [27]. Even though the first drone came in 1916 it took several decades before development impelled. The drone in mind throughout this thesis is called a quadcopter. The quadcopter is inspired by helicopters and uses four rotors in order to stay in the air. This gives the quadcopter the ability to perform vertical takeoffs and landings also called VTOL [18]. The concepts in this thesis are therefore created for this purpose, but can easily be used in other applications as well.

One of the main advantages of a drone is the ability to fly. When a drone gets altitude, there are fewer obstacles blocking for communication, it is easier to get a good overview and it is faster to travel from point A to point B. These advantages make a drone very suitable for a wide range of tasks such as photography, surveillance, transportation, rescue missions, etc. In all the examples mentioned, more drones operating at the same time could be beneficial. In a rescue mission, several drones can be sent out with cameras or other searing equipment and get a good overview a lot faster than a human search team. Several drones could, therefore, mean the difference

between life and death.

Several drones could also be used for surveillance. They could watch a perimeter instead of guards walking around, or places like California could use them to look for wildfires. There are approximately 133 million dekars of forest in California, and their biggest threat is wildfires [31]. By using several drones with for example infrared cameras, they can fly around and discover the fires when they are small enough to still be extinguished. With today's technologies, operating several drones at a time is possible. But in order to let the drones be as efficient as possible, they will need to be in continuous operation. For a drone, this means that it will need to replenish its energy. If there are only a few drones, this could, in theory, be done by humans physically changing their battery themselves, but when the number of drones increases and they have a continuous operation it would be easier to use a drone airport that changes the battery automatically.

This thesis is written for the Norwegian defense research establishment (FFI), and in their previous work, they have created what is called a battery changer robot [29]. The purpose of a battery changer robot is that the drones will find the robot when it needs to change its battery, land on it or beside it and the robot will change the battery. When the battery is changed, the drone can continue its operation. When this thesis talks about a drone airport, it will be referring to a battery changer robot or something equivalent. FFI is also working on operating several drones in what they call a drone swarm. Figure 1.1 shows an image of a test done with their drone swarm in 2018. The arrows illustrate how the drones can communicate.

Since drone airport is a pretty fresh contribution to any field, there seems to be no previous work to look at when trying to optimize it. But regular airports are quite similar and can to a certain extent be used as a reference system. As a regular airport, a drone airport is a place where the drones can land and get service performed. Now say that we include several drone airports. They can be spread out in an area, or be placed side by side. This system of airports now has several places a drone can land. If this system now had a centralized control, like a control tower, the system would be very much the same as a regular airport.

The air traffic control (ATC) is responsible for optimizing the scheduling for a regular airport. They are divided into several teams, illustrated in figure 1.2, that focus on different areas of a flight [2]. One team focuses mainly on aircrafts high in the air before a second team is responsible for the approach to an airport. When the aircraft touches the ground, the control is handed over to the control-tower stationed at the airport. Their focus is mainly taxiing the aircraft at the airport [13]. One of the most important elements for a system like this to work is the communication between the teams and of course all communication with the aircraft. Without communication, aircrafts don't know where to go and when. This type of control is called centralized control. The ATC acts as the centralization and tells

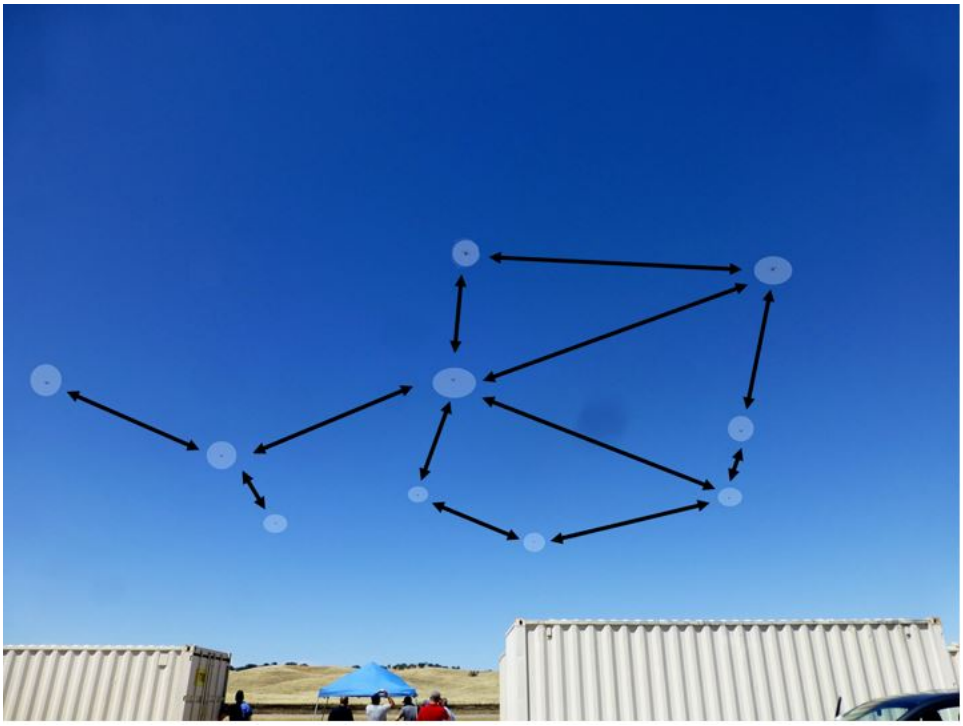

Figure 1.1: Typical FFI swarm (Sondre Engebråten, FFI 2018)

all aircrafts in the system what to do and when. These decisions are based on information from aircrafts, airports, and radars. As a guidance tool, the ATC can in some cases use a computer program that takes all information in hand and calculates a suggested order for which the aircrafts should land at the airport [37].

When creating a drone airport system, it is desirable to make the system as robust as possible. When looking at the regular airport there are two elements that are critical:

1. Centralization
2. Communication

The first point is an issue since all agents depend on this one unit. If the centralized control unit is removed from the system, especially if it is unsolicited, the whole system will crash. The drones will no longer know what to do and when. Centralization also becomes a problem when the second point above, communication, becomes restricted. Whether reduced communication is intended or not, it means that information will be difficult to maintain. Communication can often be the cause of trouble and is, therefore, together with centralization, the main reasons for looking at other ways to optimize a drone airport system.

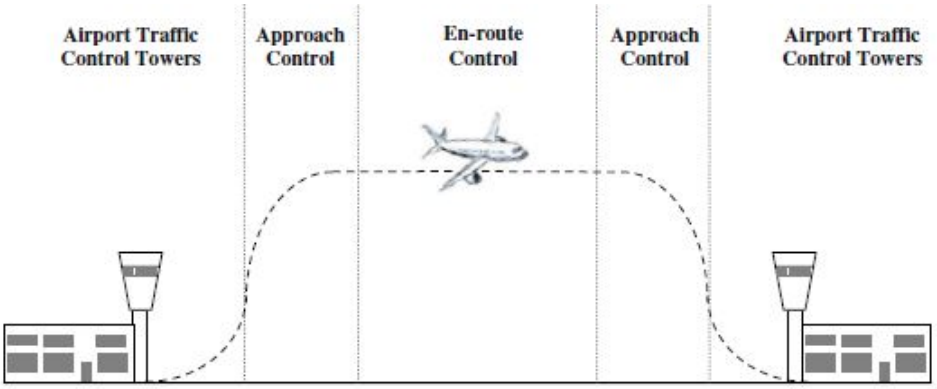


Figure 1.2: Illustration of the different ATC teams and where they operate (page 117 [5]).

In order to find the perfect optimization method for a given problem, it is important to know what and why the system is optimizing. Therefore we need to know what is important in a drone airport. As mentioned above the system will be autonomous, and therefore needs a method that can operate without needing human assistance. The reason for having an autonomous system was due to the possibility of having a very large system. With this large system, it is also beneficial if it is easy to vary the size. It can be either to add more agents or remove them. Since the system will vary its size, and as mentioned above that centralization makes the system more vulnerable, it can be advantages to make the system decentralized. Finally, maybe the most important, the system cannot be dependent on full communication. It must be able to work with full, limited and no communication. Summed up we get a list of attributes that an optimization method needs for it to be used in the drone airport system:

1. Applicable to autonomous systems.
2. Easy to vary size.
3. Decentralized control.
4. Independent regarding communication.

Since the problem at hand concerns a swarm of drones, it seemed interesting to find an optimization method within the field of swarm intelligence (SI). SI is a field inspired by insect and animal societies, and is used for distributed problem-solving [8]. SI has several interesting optimization methods, but in this thesis it is the "response threshold" method that will be tested. This method is inspired by how ants react to stimuli from other ants in terms of performing a task or not [8]. By letting the communication

between the drones be the stimuli and the airport selection be the reaction, this method seem to both fulfill the list above, and be possible to implement.

## 1.2 Goal of thesis

Since the issue regarding a drone airport is such a new research field the first goal is to create a good framework for running the experiments, including creating a good simulation tool to perform experiments. The tool must include a good way to vary the amount of communication, and it must be possible to test different optimization methods. This goal also acts as a prerequisite for being able to achieve any successful attempts on goal 2.

The second goal concerns finding an optimization method in order to optimize the airport selection done when a drone needs to change its battery. Since there are no existing methods to compare the new contribution with, this goal includes creating good benchmark methods as well as creating a method which can operate with decentralized control and limited communication. As mentioned in section 1.1, the method tested in this thesis is inspired by response threshold.

The two main goals can be summarized by the research questions bellow:

1. Is it possible to develop a suitable simulation tool to the drone swarm system where it is possible to implement different optimization methods which vary in degree of centralization and communication?

2. Is is possible to optimize a decentralized set of drone swarm airports with a method inspired by response threshold?

## 1.3 Outline of thesis

Chapter 2 gives a brief introduction to the theory of the methods used in the thesis, while chapter 3 shows how the methods are adapted for the specific problem at hand and explains how the methods are implemented in order to simulate, and gives a detailed explanation of how the simulation program works. Chapter 4 starts with presenting the specific choices regarding parameters and configurations for the simulations, before presenting the results and analyzing them. In chapter 4 there is also a comparison of the methods. Chapter 5 contains a discussion and summary of the thesis before suggesting possible future work.

# Chapter 2

# Background

## 2.1 Mathematical optimization

Even though a regular airport is optimized by humans, the decisions are often made with help from a computer program like ARAM [37], which uses all information available to suggest an optimal solution regarding the scheduling. When trying to solve complex problems it is often beneficial to look for the optimal solution. The most obvious way of finding the optimal solution is to use a mathematical approach. By using what is called Mathematical optimization it basically means to find the best solution to a mathematically defined problem [34]. This way of optimizing is used in various applications in many research fields. When talking about mathematical optimization of a problem that can be represented as an objective function, one often thinks of two main methods depending on the problem to be solved: linear-programming and nonlinear-programming. Since linear problems are much easier to solve, linear programming is often used whenever it is possible. Both methods are used in a wide range of applications in most of mathematics and natural sciences, as well as economics and statistics [35].

One of the main problems with mathematical optimization is that in order to find the mathematical best solution, the optimizer needs a lot of information about all agents. In a drone airport system, this means that it will need to know where the airports and drones are, how many drones that are at each airport and the battery status of all drones at all time. Communication is therefore crucial.

Given that we can look past the fact that communication is needed, there are two other issues with mathematical optimization. The first is that it is sensitive to size. It can handle that the size changes, but if the system becomes too big, the number of calculations needed to evaluate the entire system fast enough becomes too high for any computer to handle. In other words, the system doesn't have time to find the optimal solution before

it is too late. The second issue is that mathematical optimization uses centralized control. All calculations are done centralized in order to find the optimal solution. When crosschecking our list over attributes wanted in our preferred optimization method for the drone airport system, there is no doubt that the mathematical approach is inadvisable.

Even though the mathematical approach seems difficult to use to optimize the entire system, it can give a good idea of how well the system actually can operate. This thesis will therefore use linear programming as a benchmark to show how strong this optimization method is.

### 2.1.1 Linear programming

Linear programming saw the day of light in 1947 when George B. Dantzig was asked by his Pentagon colleagues to figure out a faster way for the military to plan the deployment of forces and equipment [17]. As a result Dantzig presented the basic linear-programming problem as we know it today:

$$\begin{aligned} \text{Maximize: } & \sum_{j=1}^{n} c_j X_j && (2.1) \\ \text{Subject to: } & \sum_{j=1}^{n} a_{ij} X_{ij} = b_i && (i = 1, ..., m) \\ & X_j \geq 0 && (j = 1, ..., n) \end{aligned}$$

In equation 2.1 the objectiv is to maximize the objective function by varying the vector variale $X$. $a$, $b$, and $c$ are vectors of known coefficients.

Linear programming can be applied when optimizing a linear objective function where the constraints are linear equalities or linear inequalities [22]. A linear problem can have as many variables as needed, but the complexity increases with the number of variables. There is also no upper boundary for the number of constraints. To easier understand the properties of a smaller linear problem (three or fewer variables), the constraints can be visualized in a graph where the variables represent one dimension each. Each constraint can be represented as a line in the graph and knowing that this is a linear problem, the lines will always be straight/linear. In both linear and non-linear programming, the objective is to find the best solution among all possible. The set of possible solutions is called a feasible region, and a problem needs to have a feasible region in order to be solved. When plotting all constraints in the graph, they form the feasible region needed. This effect can be viewed in figure 2.1. In the problem represented in the figure, there are at least four constraints since there are four edges to the yellow feasible region. Since the figure is in 2D, it means that there are

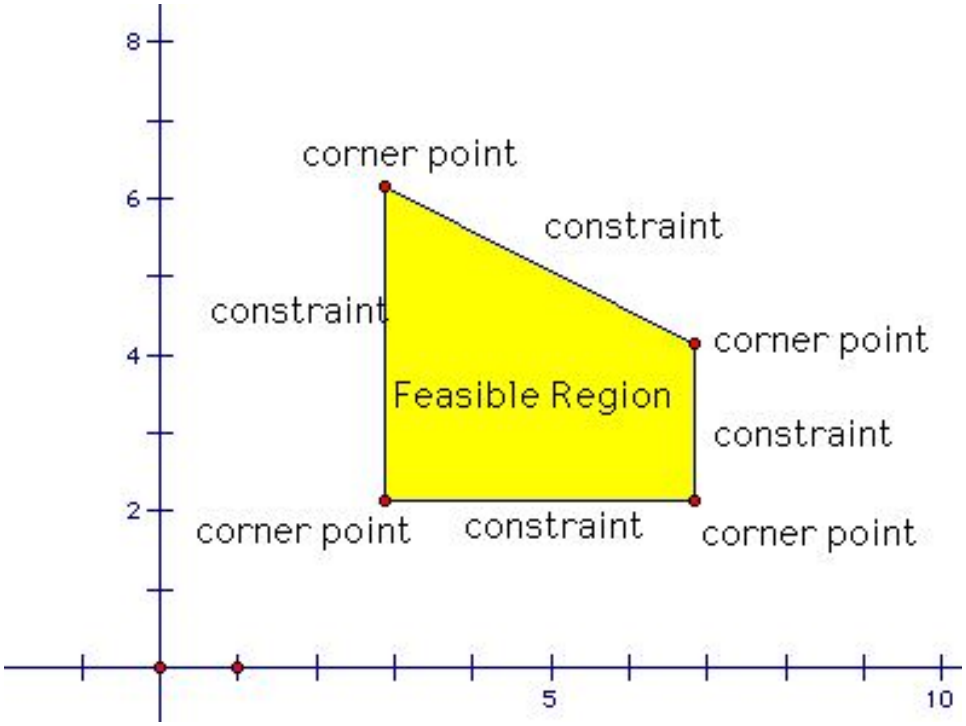


Figure 2.1: Illustration of the feasible region of a linear programming problem [38].

2 variables to the problem. In figure 2.1 there are also four corner points. In linear programming, the solution will always lie in one of the corners or vortexes [39]. Therefore drawing the graph as in figure 2.1 is often used as an effective way of solving the problem.

### 2.1.2 Non-linear programming

Like in linear programming, non-linear programming also tries to optimize an objective function. The biggest difference is that non-linear programming deals with problems where the constraints are nonlinear [4]. Non-linear programming is often used since it can be difficult to adequately represent realistic problems as a linear problem [4]. Like done in linear programming, one can also illustrate the feasible region of a nonlinear problem in a graph. Figure 2.2 shows a problem with two variables: x and y. To the problem, there are two visible constraints that have a curve, and form the feasible region located in between (illustrated in blue). In a nonlinear problem, the optimal solution does not necessarily lie in a corner or vortex, and can therefore often be more difficult to find. In figure 2.2, the optimal solution lies in the tangency between the feasible region and the diagonal tangent [28]. The tangent is decided by the objective function.

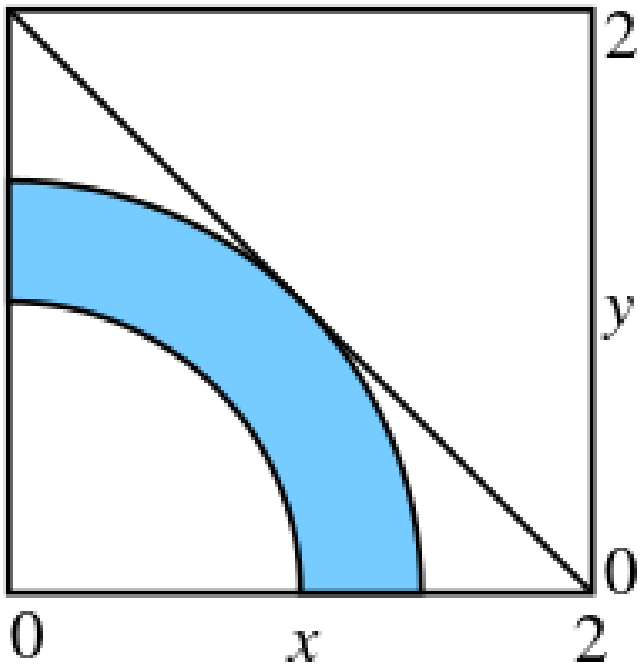


Figure 2.2: Illustration of the feasible region of a non-linear programming problem. [28]

Since the problem related to the drone airport system is possible to convert into a linear problem, it will be linear programming that will be used in order to understand how well one airport can be optimized.

## 2.2 Multiagent systems

Multiagent systems come to mind as a relevant method for solving complex problems where there are several agents making autonomous decisions regarding their own state as well as interacting with one and other. Multiagent systems have been studied at least since the 1980s [43] and is a collective term for several systems including game theory and swarm intelligence which both focus on understanding the effect on a system when agents interact [30] [6].

The term agent is used in multiagent systems as a player or individual which affects the system, and figures out itself how to satisfy its design objective [43]. The agent can vary in terms of complexity. As seen in figure 2.3 a game theory agent often needs to be strategic/complex in order to fully understand and interpet the game. The swarm intelligence agent is on the other end of the complexity scale, and is called reactive. This agent usually react to something in the environment. Stratigic and reactive agents are

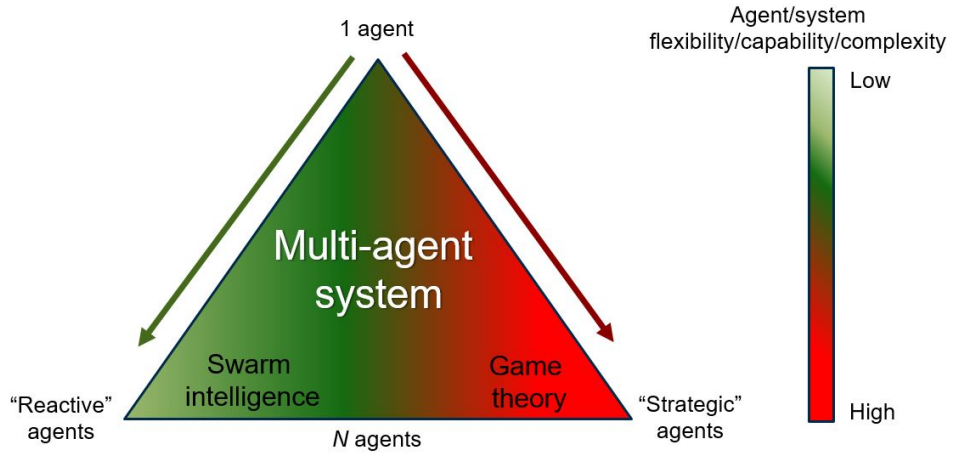


Figure 2.3: Illustration of how the agents differentiate in terms of complexity in a Swarm intelligence system and Game theory. (Illustration by: Jonas Moen, FFI 2019)

only the outer points in figure 2.3. The agents can also be a combination of the two, and is often created as complex as needed for its use.

### 2.2.1 Game theory

In game theory the game is a description of strategic interaction between two or more agents or players as they are called in game theory [30]. When working with game theory there are usually two assumptions made of the players: they are rational, and intelligent [26]. When a player is rational and intelligent it will always pursue the players own objectives by choosing the best action given their perception of a given state [15].

Game theory is often a theoretical way of understanding what happens when decision-makers interact and use mathematics to express this [30]. Dantzig, who was the creator of linear programming, explained in 1951 how linear programming is equivalent to a zero-sum game with two players [12]. In 2012 Adler filled the gaps in Dantzig's study regarding "the relationship between the Minimax Theorem of game theory and the Strong Duality Theorem of linear programming"(page 1 in [1]), thus completing the connection between the two.

Though game theory clearly has many of the same advantages as linear programming when it comes to finding optimal solutions, it will not be used in this thesis due to the same issues regarding communication in order to obtain the perception of the whole system.

### 2.2.2 Swarm intelligence

Swarm intelligence, which is considered an artificial intelligence discipline [6], uses the basic idea that relatively simple individuals, together create a more advanced system. The individuals make decisions based on the environment, which often includes the behavior of other individuals. This has resulted in many different optimization methods like "ant colony optimization" (ACO) and "particle swarm optimization" (PSO). The PSO is based on bird flocks and fish schools and uses social-psychological principles to find an optimal region in a search space [20]. The ACO is, as the name states, inspired by ants. The method is designed to mimic ants' search for food and resources and can be used to solve combinatorial optimization problems [8]. Insects, in general, became a very interesting research topic due to their incredible division of labor. In Wilsons observations [40] [41] [42], he found that the ants responded differently to stimuli and therefore divided the labor based on how they responded to a given stimuli. To easier understand these observations, Bonabeau et al. [7] developed a model that relies on response threshold [32] [33]. This model shows that each individual has a response threshold per task. This way, the ants who are supposed to find food, will respond on less stimuli associated with this task than other ants [8].

When calculating the response in a response threshold system, bonabeau et al. [7] [8] present two equations that can be used. Equation 2.2 and 2.3 calculate the probability of responding to stimuli $S$, given response threshold $\Theta$. In equation 2.2, $n > 1$ affect the steepness of the probability, and determines how quickly the probability goes from 0 to 100%. For equation 2.3, the threshold itself ($\Theta$) decides the steepness of the probability.

$$T_\Theta(S) = \frac{S^n}{S^n + \Theta^n} \tag{2.2}$$

$$T_\Theta(S) = 1 - e^{-s/\Theta} \tag{2.3}$$

Figure 2.4 shows equation 2.2 plotted with different values of theta. As one can see from the plot, the probability goes from 0 and converges towards 1, meaning that the more stimuli present, the greater chance for responding. By looking at the figure it is also easier to understand that the threshold is not a boolean determining whether one needs to react or not, but is used in the calculation to decide when the probability of reacting should exceed 50%.

Response threshold is used in several applications, like feature selection in mathematics [16], task allocation and search planning on a UAV[21] and adaptive behavior in robotics [10] to mention a few. There are also many researchers that have implemented responses based on some threshold without necessarily using the equations mentioned above from the studies from Bonabeau et al. A few examples are Mann and Poore who uses response

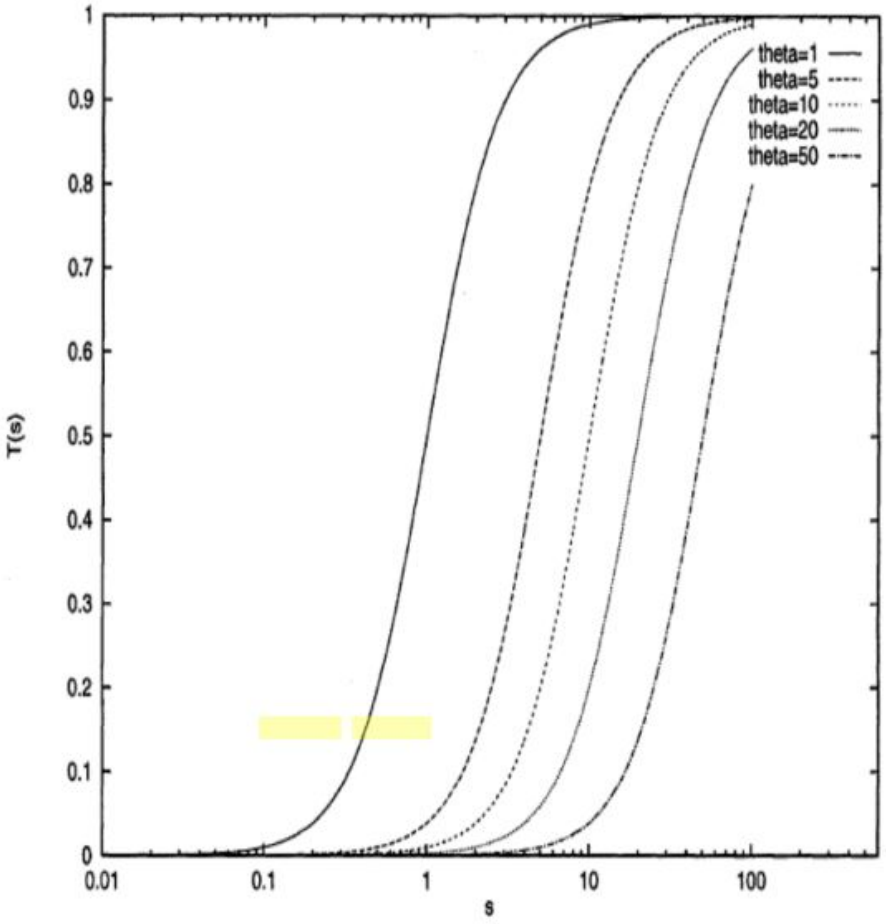


Figure 2.4: Semi-log plot of response curves to equation 2.2 where $n = 2$ and $\Theta = 1, 5, 10, 20, 50$. Figure from Bonabeau et al. [8] page 114.

threshold in the microcontroller of a pacemaker in the field of medicine [23], or Mansfield et al. who look at how the response threshold adjusts in the human brain when switching tasks [24].

The idea of responding differently to stimuli or signals based on your own observation and state is what makes response threshold an interesting method to use in our drone airport system. Comparing with the list of attributes wanted for the optimization method, response threshold seems to be a match. First of all, it is very applicable in an autonomous system, and secondly it is not sensitive to size nor variable size. Regarding point 3, decentralized control, a multiagent system is perfect since all agents control themselves. Last but not least the amount of communication is very scalable in a response threshold method. Therefore, the method easily checks off every point on our list over attributes wanted.

## 2.3 Performance in swarm robotics

Even though swarm intelligence is based on insecet and animal behavior, it is more and more used in engeneering problems [36]. The system at hand in this thesis which consists of drones and airports can be called a swarm robotic system. Swarm robotics is basicly the study of how physically embodied agents with local interaction can create a more intelligent system then by them selves [14].

The performance in swarm robotics is often dependent on the density of agents in the swarm [19]. A good example of this is something called bucket brigades. Bucket brigades is a way of transporting objects from one point to another using a form of task partitioning [3]. Instead of all agents traveling back an forth with the objects, they get in a line, and pass the objects inbetween one another. An illustration can be seen in figure 2.5 (a) and (b). (a) shows that when there are to few agents they can not create a bucket brigade, while (b) shows how they create lines when they can.

In figure 2.5, (c) and (d) show the overall performance given the number of agents. Many swarm robotic systems have this performance funciton, and can often divide it into the 4 regions shown in (d) [19]. Throughout this thesis, the term saturation will appear many times when referring to how many drones that are at an airport. In figure 2.5 (c) and (d) the saturation point is the maximum value. The point where adding or removing a drone will decrease the performance. In other cases saturation can be defined as when a state or quantity is adequate in terms of collecting data [25], but it also seems that the determination of saturation is in a lack of explicit guidelines [9].

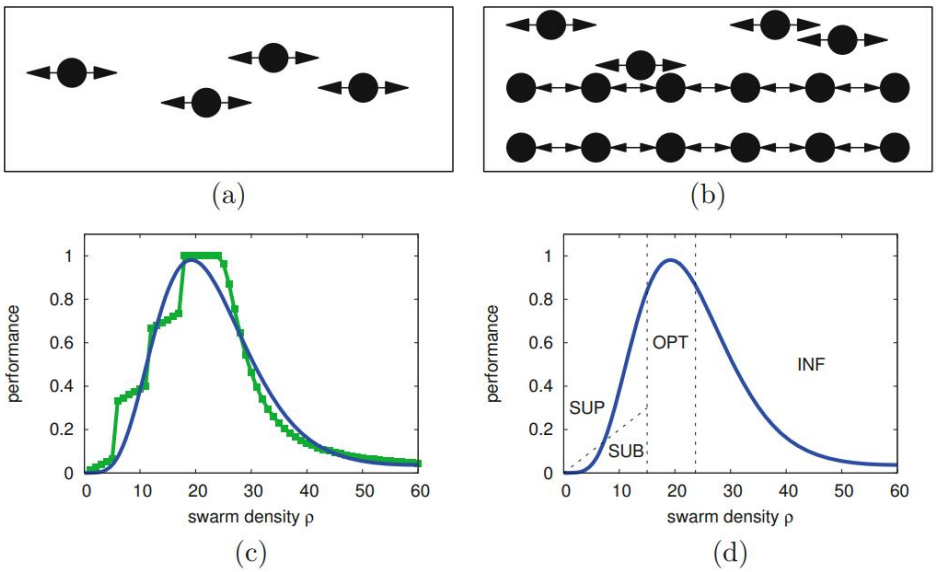


Figure 2.5: Bucket brigade example for swarm performance and typical swarm performance function over swarm density. (a) Bucket brigade, 4 robots (b) Bucket brigade, 16 robots (c) Bucket brigade, performance. (d) Swarm performance showing four regions, SUP: super-linear, SUB: sub-linear, OPT: optimal, INF: inference (Figure and caption from page 10 in [19])

# Chapter 3

# Method & Implementation

This chapter will explain the implementation leading up to the experiments presented in chapter 5. The chapter will first explain the simulation strategy and how this affects the simulation, and then give a more detailed explanation of how the simulation works and how the methods are implemented.

## 3.1 Simulation strategy

A system like a drone airport system is very dependent on time. Time affects when the drones arrive at and leave an airport, which again is dependent on the time an airport needs to give service to the drone. The amount of time a drone needs at an airport combined with how long the drone can stay airborne, again decides how many drones the system can handle, and so on. Since time is of such essence it seems natural to create a simulation system that takes time into accounting. This thesis is therefore built up as an explicitly time-dependent simulation system, where clicks simulate time. By doing so it is possible to build up the system as close as possible to a real-life drone airport system, and therefore minimizing the reality gap.

When finding a suitable programming language for the simulations, there are several good options. For this thesis, the simulation has been done in Python due to its powerful packages for mathematical operations and visualization combined with the ability for object-oriented programing. This makes Python a powerful tool while still allowing rapid development. In order to make the system as realistic as possible, the system uses Pythons object-oriented ability to split the system into smaller parts. The objects created in the simulation correspond to what they are in real life. The implementation will be described in detail below.

## 3.2 Visualization

When running simulations, especially click based simulations, it can be advantages to view the simulation to be sure that everything is doing what it is supposed to do. As mentioned above, one of the advantages of using Python is a good visualization package which makes it simple to view the simulation. The package used in this thesis is called Pygame and provide the drawing methods necessary. In figure 3.1, an example of a simulation is shown. This simulation is rather small but illustrates the visualization tool created for this thesis. The light grey area is the drone workspace. This area specifies where drones can be. This means that for this thesis the drone workspace is limited. This decision was made since many of the applications thought for this system have limited search or surveillance areas. Inside the drone workspace, there are five green dots. These dots represent the drones. Each drone has a number and a bar overhead which correspondingly represents the id and battery level for that drone. In figure 3.1, drone 1 is highlighted, and it is therefore possible to see all the target points that drone 1 has. The targets are visualized by flags. The figure also has two airports represented by circles. There is one far right which is green, meaning it is available, and a red one far down indicating that it is occupied. One can see that drone 3 is in the center of the airport furthest down, which again confirmes that the airport is occupied. A black circle surrounding the occupied airport illustrates how far into the battery change the airport is. When this circle completes around the airport, the drone will take-off and continue its mission.

Since drawing and rendering the graphical user interface viewed in figure 3.1 needs a lot of computational power, it slows down the simulation. Therefore it is implemented a button to turn off the animation, which allows the simulation to run much faster. There are also buttons to add more airports or drones if it is wanted to adjust the size of the swarm. In the top left corner, there is a counter for the clicks. This counter is the clock of the system and shows how far the simulation has come.

## 3.3 Simulation tool

In order to visualize as in figure 3.1 there need to be objects that can be drawn. In this case, we have the two objects already mentioned above, a drone and an airport. Since this is a simulation, centralized control is needed in order to control the system, and make all the agents move. In this layout, the centralized control is called “simulation core”. The last object is the control tower which acts as a control tower in a regular airport.

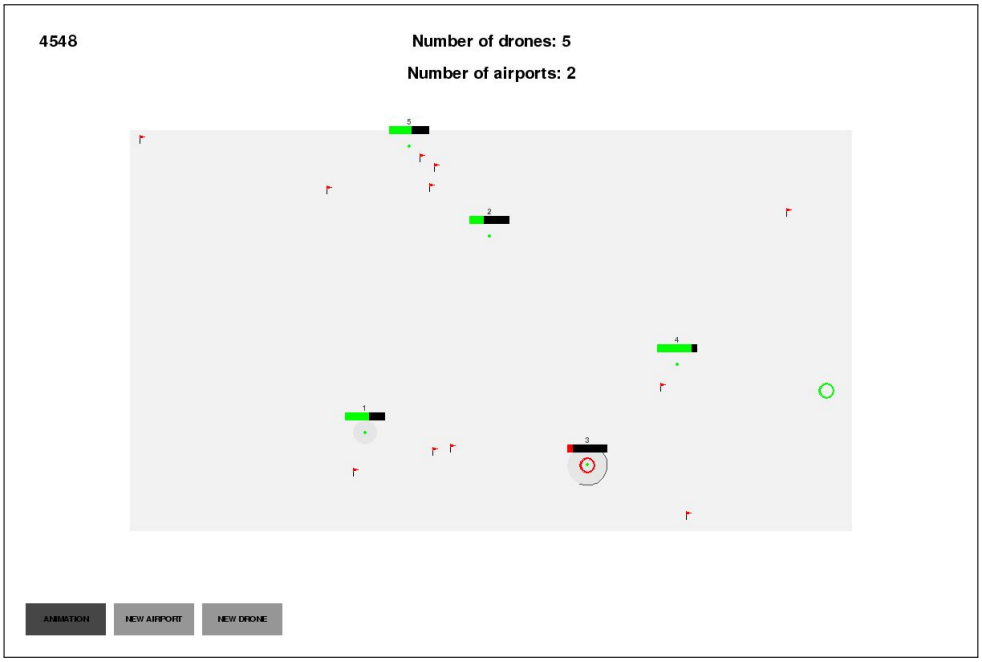


Figure 3.1: Illustration of the graphical user interface created for the thesis.

### 3.3.1 Simulation core

Although one of the main goals is to decentralize the system, one needs to centralize the simulation since everything is running on one computer. This means that the simulaiton core tells all drones to move one timestep, and then tells all airports to update. The simulation core then runs the optimization corresponding to the mode it is running, and follows this loop until the simulation ends.

### 3.3.2 Control tower

The control tower object operate as the centralized control unit when simulating benchmark systems. The drones can communicate with the control tower in order to get infomration reagarding which airport it should go to.

### 3.3.3 Parameters

To run a simulation like in this thesis there are several parameters that are important, and that needs to be easy to change in order to find the best configuration. Therefore, there is a parameter file to the simulation. In this file, one can find information on how long it takes to change a battery, how fast the drones should fly, how much communication is possible, etc. Essentially, all parameters that can be tweaked can be found here.

| Drone state | Description |
|---|---|
| Active | This state is when the drones are active on their mission. This means that the drones are flying from target to target simulating waypoints. In this state, the drone has a given speed and calculates its heading to be towards the next target. In this state, the drone also constantly checks if it has to change the battery and therefore enter the next state: Approach. |
| Approach | When the drone enters the approach state, it means that it has calculated that it needs to approach the airport due to low battery. As a safety measure, the drone needs to be at the airport when it has 15 percent battery left. The approach state has its own speed defined in the variable file, and the drone calculates the heading towards the airport. |
| Landed | When the drone arrives at the airport, it enters the landed state. This means that the speed is set to zero and that the drone enters the service queue at the airport. When the service is finished, the drone again enters the active state. |

Table 3.1: Description of the different states a drone can be in.

### 3.3.4 Drone

There are several parameters that need to be implemented on the drone. The most obvious is the position. The position corresponds to a GPS coordinate in the sense of having a latitude and longitude. This parameter decides where on the map it is, and can therefore easily be used to calculate the distance to other points. For these simulations altitude has been ignored, since it is believed to not affect the results. The only time altitude becomes interesting is when the drone is to land and take-off from the airport. By saying that these scenarios are time spent at the airport, we can increase the time it takes to change the battery, and therefore include this time in the simulations without explicitly including altitude.

In order for the drones to change position, they also have speed and heading. Both the speed and heading is given by which state it is in. There are mainly three states which are explained in table 3.1. Figure 3.2 show the state transition diagram connected to the table.

As mentioned above, the control tower tells the drones when to move. This is possible since the drone has a function called “drive”. In this function, the drone always runs a status check, which checks if everything is all

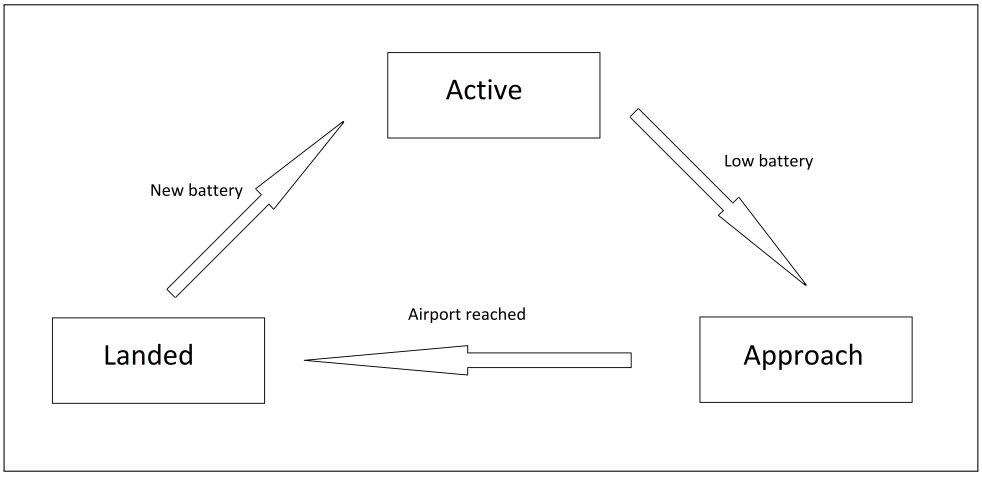


Figure 3.2: Illustration of the state transition diagram for the drone.

right with the drone, and updates a boolean referring to the status. The status check also calculates based on the distance to the airport if the drone needs to enter the approach state. Given that everything is all right with the drone, it can then act correspondingly to which state it is in. Depending on which mode the simulation is running, the drone also has other functionalities within the drive function. The different modes will be described in a later subchapter accompanied with how they are implemented.

### 3.3.5 Airport

Like the drone, the airport also needs several variables in order to operate. Since the airports are stationary they do not have a speed or heading, but they need to know their position and how much time they should use on a battery change. How much time it takes to change a battery decides a lot in terms of how many drones an airport can handle. Equation 3.1 is the mathematical way of figuring out how many drones an airport can handle based on how much time it takes to change a battery, how large the battery loss per time step is, the average percentage in a battery when a drone receives it and how much battery the drone has left when landed. If one has a greater system with several airports, the equation (3.1) can be multiplied by the number of airports, or calculated for each airport and added together.

$$N_d = 1 + \frac{(\tilde{b}_D - b_A)T_L}{T_{bc}} \tag{3.1}$$

$$T_L = \frac{1}{b_L}$$

$N_d$ : Number of drones which saturates an airport
$\tilde{b}_D$ : Average battery percent when departing airport
$b_A$ : Battery percent at arrival
$b_L$ : Battery loss per click
$T_{bc}$ : Number of clicks to change battery

By doing this calculation one can quickly find out which state the system or airport is in when it comes to saturation. The system can either be under-saturated ($N << N_d$), saturated ($N \simeq N_d$) or over-saturated ($N << N_d$) where $N$ is the number of drones in the system. Each state has both advantages and disadvantages, which can be viewed in table 3.2.

The airport class also keeps track of which drones are at the airport. This means that the airport can have several drones at the airport at once. In the simulations done for this thesis, it is assumed that an airport can have an infinite number of drones in a landed state, and that the airport either drives around in a specific area doing service on the drones or picking the drones up and placing them where the service is being done. In the simulation, this is simulated by the drones landing at the same place even though there is another drone present already.

When the control tower tells the airport to update, the control tower calls a function called "runClick". This function updates all information inside the airport. Just like in the drone, the airport also runs a status check. This check mainly updates whether the airport is occupied or not. When an airport is occupied, it means that it has drones at the airport that needs service. After the status check, given that it is occupied, the airport keeps track of the counter used for the battery change.

### 3.3.6 Battery change

When a drone has landed at an airport, the airport immediately starts a counter, given that it is available, which simulates a battery change. When the counter reaches the amount corresponding to how long time a battery change takes, the airport creates a new battery and gives it to the drone. The battery percentage on the drone is randomly chosen between 80 and 100 percent. The randomized pick uses a uniform distribution. This feature is included to simulate that the batteries have diverse conditions, and often vary in terms of starting percentage and how fast they empty. By introducing the unpredictability of how fast the battery empties into the starting percentage, the thesis assumes that the battery loss per time step is fixed.

| | Pros | Cons |
|---|---|---|
| Under-saturated | • No waiting time for the drones. | • Does not utilize capacity.<br>• The airport has a lot of downtime. |
| Saturated | • The system is very effective due to little waste of time both from the airports' point of view, and also the drones (given that it is optimized). | • Can be very computationally heavy to optimize, and therefore not very scalable. |
| Over-saturated | • No need to optimize, just need to "pick" one of the drones that are waiting.<br>• You know that the system is working 100%. No time where the airport has nothing to do. | • The drones need to wait more often.<br>• System is very vulnerable to errors, since the system is always "overbooked". |

Table 3.2: Advatages and disadvantages regarding the different stages of saturation.

### 3.3.7 Communication

When it comes to communication, there are two interesting dimensions. The first dimension is the distance. This corresponds to the strength of communication. When talking about limited communication in this dimension, it means that a drone can only talk to other drones within a limited distance. When achieving full communication, the drones can talk to all other drones regardless of the distance in between. The second dimension is bandwidth and determines the amount of communication. With limited communication in this dimension, it means that a drone can only pass a limited amount of information to other drones. When talking about full communication in this dimension, it means that a drone can give all information to other drones.

Since limited communication is one of the reasons why swarm optimization is being tested, the simulation also needs to be able to vary the amount of communication. The first dimension mentioned above is implemented by creating a grid over the drone workspace. When a drone enters a region of the grid, the drone is put on a list corresponding to this region. This list then represents which drones can communicate with each other. When a drone leaves an area it is then removed from the list. The simulation core is the object which keeps track of all the lists. Although this way of simulating communication doesn't mirror the reality perfectly, it is an effective way of testing limited communication without having all the drones check the distance to all the other drones in every time step. By reducing the size of the regions in the grid, one simulates less communication in terms of distance. Likewise, the communication distance increases by increasing the size of the region.

Figure 3.3 illustrates how the grid works. In the figure there are 4 drones. Drone 2 and 3 are in the same region and can therefore communicate. Even though drone 1 is very close to drone 2 they can not communicate since they are not in the same region.

The second dimension is simulated by adding information to the list mentioned above. When a drone enters a region in the grid and is appended to the communication list, instead of simply appending the drone id, one includes the information which is to be shared. All other drones in the same region can then update their internal database when new information enters. To simulate limited information in terms of how much information is shared, the drone only adds limited information to the list.

## 3.4 Simulation setup

In order to understand why the different optimization methods were chosen, one must understand what was important to test during the simulations. There are several parameters to a system like this in terms of setup and configuration. One of the interesting ones is where the airports are placed.

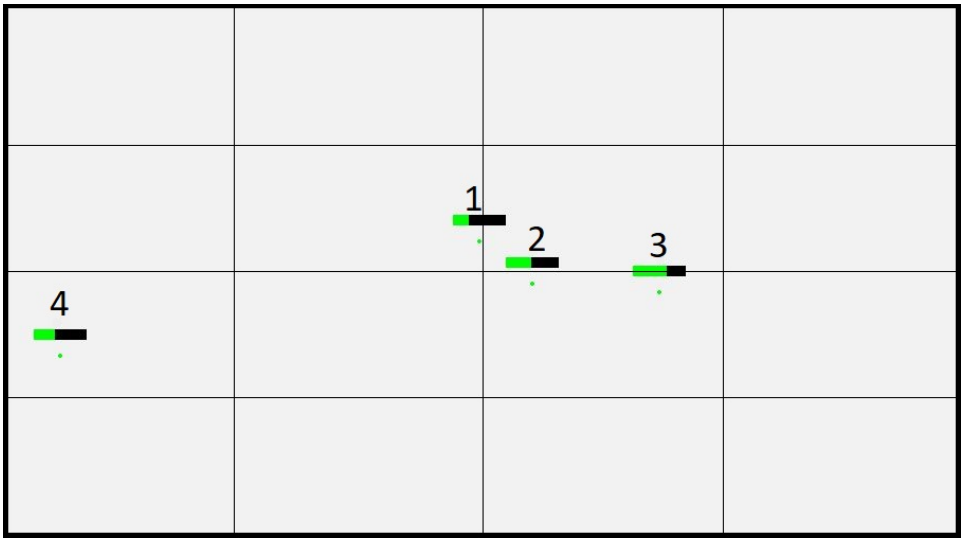


Figure 3.3: Illustration of the communication grid. Each square represents a region where the drones in the same region can communicate with one another.

There are several possible strategies when placing the airports: All can be placed at the same location either inside or outside of the drone workspace, or they can be spread out both inside or outside the workspace. If they are spread outside the workspace, they can have a fixed equal distance to the workspace, or they can have different distance. All the different configurations of the airports have their advantages and will affect the system in some way. This thesis has focused on two configurations of the airports, and the methods are therefore tested and created for these configurations. The first one is the most basic, and have all the airports located at the same point. In a real-life test, the airports can naturally not be located at exactly the same point, but they could be so close that they appear to be. The second configuration tested in this thesis has the airports spread outside the workspace. In this configuration, the goal is to simulate that the airports are at different fixed distances outside the workspace. This means that the time it takes to reach an airport is the same for all drones wherever they are in the workspace, and that each airport has its own distance.

One of the other interesting parameters to look at when it comes to setup and configuration is the amount of initial information given to all the agents. More precisely, how much the drones and airports know about each other at setup. Especially for the swarm method, the behavior depends on if the drones initially know where all airports are, or if they only know about the one they started from. Both these scenarios will be simulated, and the difference in all the methods will be better explained below.

When it comes to communication there are only one of the dimensions

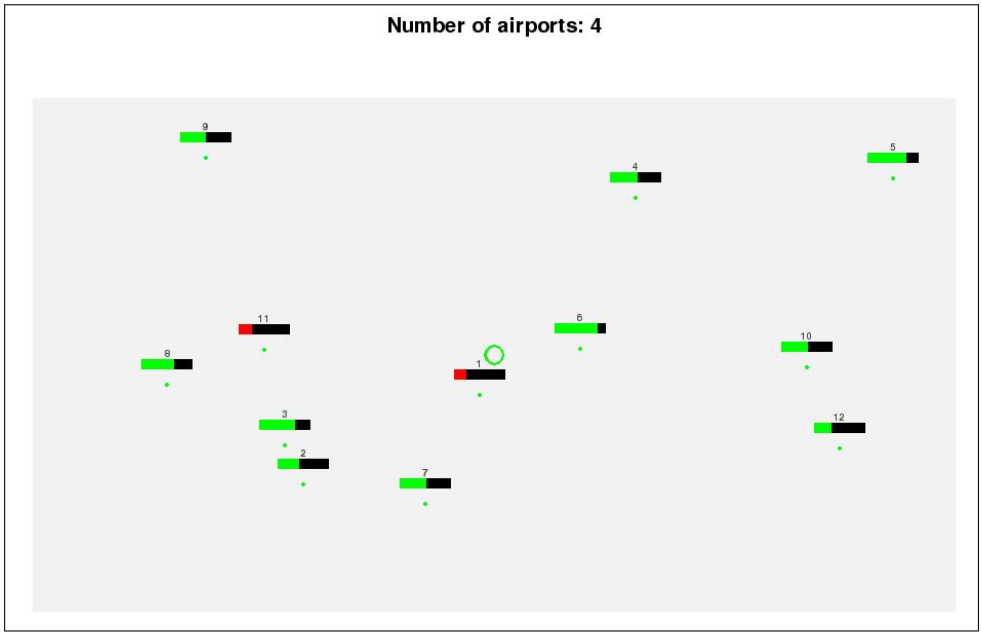


Figure 3.4: Illustration of a simulation where all airports are located at the same point.

explained earlier, the communication range, which will be varied. This is because prior to the simulations, tests have been done in order to find the information which is most important for the drones. This information will, therefore, be the information given to other drones in all simulations where limited communication is applicable. The communication range, on the other hand, will be simulated using the grid system explained above.

### 3.4.1 All airports at the same point

As explained above, the airports in this configuration are located at the same point. The reason for including this configuration is to look at the system where the distance to the airport is negligible. To make sure that distance has as little to say as possible, the airports are put in the middle of the drone workspace. Another advantage of this configuration is that it simulates a real-life application where it can be very difficult to place airports in different places in the workspace. This can be due to the lack of roads or lack of open places suitable for drone airports.

### 3.4.2 Airports spread with individual fixed distance

The second configuration keeps the airports outside of the drone workspace, and in addition, they have different distances to the drone workspace. The reason that the distance to an airport is fixed in this configuration is to

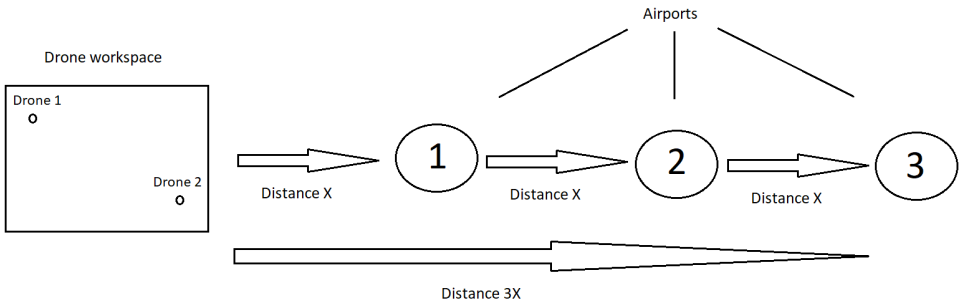


Figure 3.5: Illustration of fixed distance to each airport.

remove the effect that when a drone moves around it can come closer to one airport while it moves further away from another. This effect makes it very hard to calculate which airport the drone should go to without also calculating exactly where the drone is at any given time. Figure 3.5 shows the concept of the configuration. In the figure, there are two drones. Both these drones have distance X to airport 1 even though drone 1 is further away. The same goes for airport 3; both drones have distance 3X to airport 3. As in figure 3.5, the simulation also uses this spread in the airports. Meaning that if airport 1 is distance X from the drone workspace, then airport 2 is double the distance and airport 3 is three times the distance. For the simulations, the distance X is set to be equal to a half battery change. This way it will take a whole battery change to travel both to and from airport 1. Traveling to airport 2 and back will take 2 battery changes of time.

## 3.5 Optimization methods

The biggest differences between the methods studied in this thesis concern communication and centralization. To easier understand difference amoung the methods, figure 3.7 shows an illustration of all the methods tested in the simulations, and show how they differentiate when it commes to communication and centralization. In the figure there is a linear line, and, as illustrated, the area above the line is an area no methods lie. This is because in order to increase the amount of centralization, it is neccessary to increase the amount of communication. It is not possible to control something if there is no communication between the controller and the agent being controlled.

As can be viewed in figure 3.7, the mathematical and centralized method is, without competition, the one with most communication and centralization, and is most likely the best. In the simulations mathematical optimizaiton will only be used for one airport. This desicion was made after testing and finding how quickely the complexity increased when incresing the size of the system. Therefore mathematical optimizaiton will only be

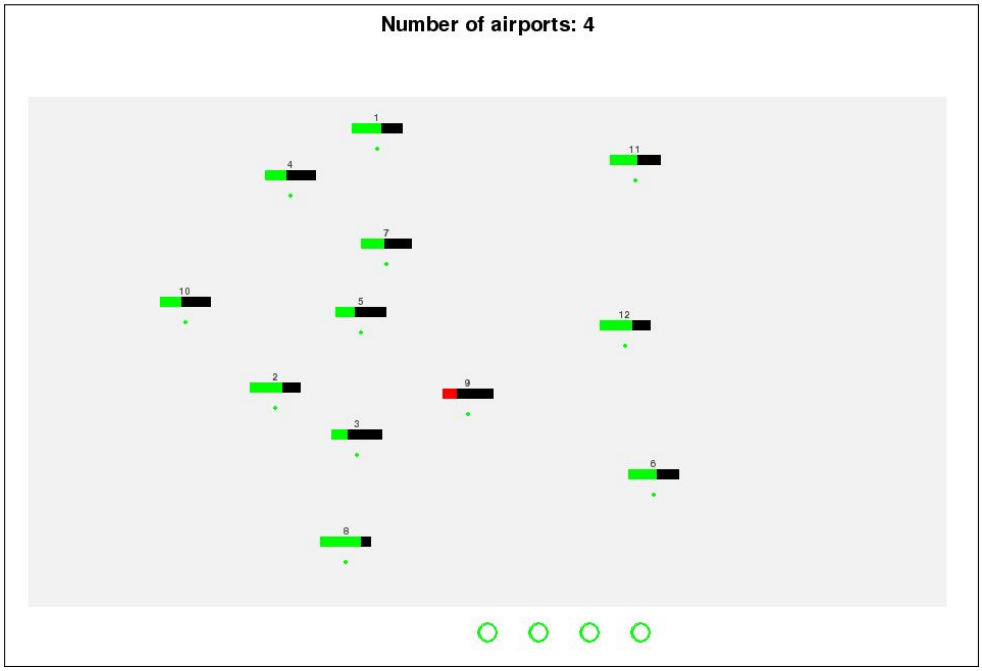


Figure 3.6: Illustration of a simulation where the airports are outside of the workspace.

used as a referance to understand how well one airport can be optimized when looking at the other methods. Instead of the mathematical optimization there will be another decision making method which uses the advantages of full communicaiton and centralization, but it is not proved to be mathematically optimal. This method will be fully explained in section 3.6. As illustrated, the swarm method can vary when it comes to amount of communication, but needs to be in a decentralized state.

In a addition to the centralized and swarm methods there is also a random method in the illustration. The random method is included due to their simplicity. The main goal of having a random method is to understand how well the system works without any form of optimization. This means that a new contribution to the field will need to be better than random in order to enhance the system.

There are two things that can be optimized in the drone swarm airport system. The first which is used when optimizing mathematically is to decide when the drone should arrive. The second is to decide which airport a drone should go to. This is only applicable when there are several airports. The second method is the one looked at when simulating several airports throughout the thesis.

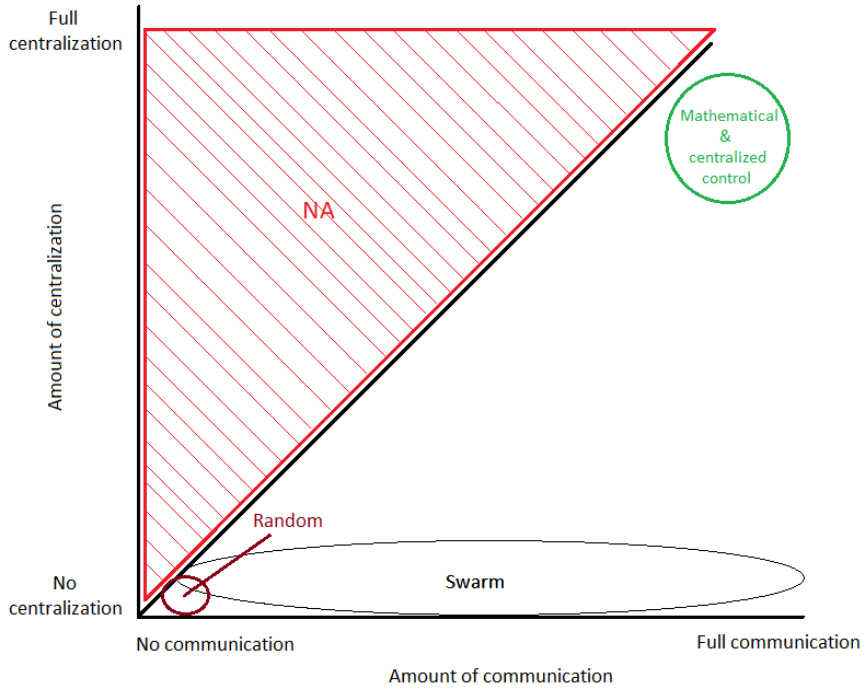


Figure 3.7: Comparison of the methods in terms of the amount of communication and centrilization.

## 3.6 Centralized optimization methods

The goal for the centralized optimization methods is to understand how well the system can perform when all information is available for a centralized controller. This means that these methods are dependent on full communication between all agents. In the introduction and background chapters above, mathematical optimization has been explained in detail since the method is used in a wide specter of applications, and is proven to give the optimal solution. The mathematical approach will, therefore, be included as a reference to see how well one airport can be optimized. When extending the system with more airports, centralized control will be represented by a method that uses less computation to optimize, because there are too many variables for the mathematical optimization method. This method is not shown to be mathematical optimal, but will be a good reference to how well the system can perform.

### 3.6.1 Mathematical optimization

The mathematical optimizaiton solution to the problem is a linear programming approach where equation 2.1 is used. As explained in the section for linear programming (2.1.1), the equation consists of an objective function and constraints which need to be defined.

#### Objective funciton

The objective function to this optimization problem is based on maximizing $T_a$ (air time) and minimizing $T_d$ (down time). By combining the two we get the objective function 3.2. The air time is all time which is not spent at the airport, while downtime is all time spent at the airport. This includes waiting till the airport is available and the service time. This can be extended to include traveling time for the drone in order to optimize based on where in the drone workspace the drone is as well, but this extention is not prioritized in this thesis. Both the air time and down time is a result of $X$, the variable vector that is to be determined.

$$\max \sum T_a - T_d \tag{3.2}$$

#### Constraints

There are only two constraints to this optimization problem. The first constraint (equation 3.3) regards the slot size, and prohibits two drones from getting a time slot too close to each other. In general, this means that a drone needs to be finished with its battery change before another drone can get a slot. A slot is the time dedicated to a spesific drone.

$$\mid X(i) - X(i-1) \mid \geq T_{bc} (\forall i \neq 0) \tag{3.3}$$

$X(i)$ : Given time slot for drone $i$.
$X(i-1)$ : Given time slot for drone $i-1$.
$T_{bc}$ : Time it takes to change a battery.

The second constraint is that the time slot given to a drone can not come before the drone is able to reach the airport (equaiton 3.4). This constraint is pretty self explanatory in the fact that a drone needs to be close enough to the airport in order to reach the slot given.

$$X(i) - T_r(i) \geq 0 \tag{3.4}$$

$X(i)$ : Given time slot for drone $i$.
$T_r(i)$ : Time for drone i to return to the airport.

### Implementation

In order to simplify the problem, there is one assumption that has been made which is believed to not affect the results. This regards the order of the incoming drones. In section 3.3.4, it was mentioned that the drone has to land when it has 15% of battery left. This means that for the drone to waste as little battery as possible it needs to land when it has 15% left. Landing before this means that battery has been wasted. Therefore, the assumption made in order to get a linear problem is that the order of the incoming drones is based on when their optimal landing time is. If drone 1 has optimal landing time 500 clicks from now, while drone 2 has optimal landing time 501 clicks from now, it means that drone 1 will come before drone 2, and the mathematical optimization method cannot change this order. This means that the optimization method only pushes around on the slot given to a drone for the battery change.

In figure 3.8 the two outer configurations are illustrated. Option 1 shows what happens if the drones land at their optimal arrival time, and then wait until the airport is available. As can be seen in the figure, drone 4 has to wait a long time in order to get its battery change. In option 2 on the other hand, the drones never wait for their turn, but they need to land earlier than their optimal arrival time. This means that they change a battery that has unused energy in it. The idea for the mathematical optimization is to find the perfect combination of some drones landing too early, and some drones having to wait in order to achieve the most efficient system.

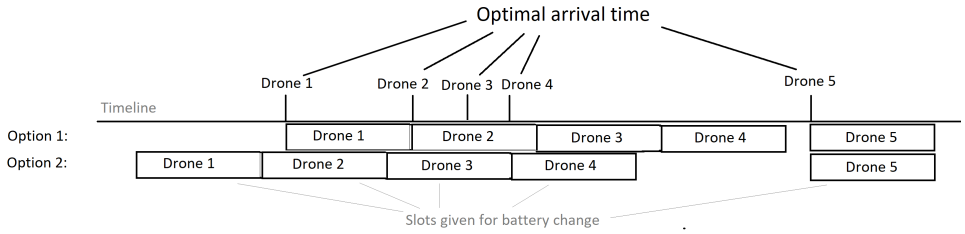


Figure 3.8: Illustration of the edge cases when distributing time slots to drones.

The mathematical optimization method is implemented as the linear programming problem described earlier. The Python package called “SciPy” gives a useful library for scientific computing. Within the “SciPy” package there is a whole package called “optimize” which gives several different optimization functions that are simple to use. For the simulations where mathematical optimization is used, it is the function “minimize” from the “optimization” package which is called. The funciton is shown below, with a brief explatation of the variables.

ret = scipy.optimize.minimize(fun, x0, method=None, bounds=None, constraints=())

- fun: The objective function to be minimized.
- x0: The initial guess which shows the dimensions the optimization problem.
- method: Here it is possible to specify which type of solver to use.
- bounds: a bound for the variables in order to reduce the amount computation.
- constraints: The constraints to the optimization problem.
- ret: The returning optimal solution.

### Objective funciton (fun)

The objective function is implemented as explained above with equation 3.2. $T_a$ can simply be found by looking at the last time a drone changed the battery, and the next time it needs to land for a battery change. $T_d$ is equal to the time it takes to change a battery, plus the waiting time at the airport before the battery changer is available. Waiting at the airport will only occur if the time slot given for battery change is later than when a drone has to land due to low battery. Since the "minimize" function minimizes the objective function, equation 3.2 is multiplied by $-1$.

### x0

The goal of the x0 argument is to tell the "minimize" function what it is searching for. In this thesis, x0 is an array of numbers. The size of the array corresponds to the number of drones in the system. The x0 can also be used to give an initial guess to what the solution should be. For these simulations, this initial guess is not in use, and the input is, therefore, an array of zeros.

### Method

Since the problem is being approached by a linear programming method, it is essential to have constraints. In order to use constraints in the "minimize" function, there are only two available methods. The first is called "COBYLA" which stands for "Constrained Optimization BY Linear Approximation".The second method, and the one used for the simulation, is called "SLSQP" which stands for "Sequential Least SQuares Programming". The main reason for using "SLSQP" is that "COBYLA" can not have bounds. This becomes a problem since it is not possible to limit the search space, and resulted in a few test runs that never ended.

#### Bounds

In the bounds argument, the goal is to inform the function of how low and high values the results can be. They work as a constraint for the calculation, while the constraints bellow works as constraints for the objective function. In the simulation, the bounds were set to (0,4000) for each drone, which means that the "minimize" function only can test solutions within this boundary.

#### Constraints

The last argument is for the constraints to the linear programming problem. The function receives a list of constraints that all have to be true in order to be a feasible solution. As explained above, there are only two constraints to this optimization problem (equation 3.3 and 3.4), but in order to implement them in the simulation, the constraints are created for each drone. This means that the number of constraints is equal to 2 times the number of drones.

### 3.6.2 Centralized control w/several airports

When simulating several airports the problem is no longer optimizing the arrival, but a question of which airport the drone should go to. When a drone needs to choose an airport, it asks the control tower/simulation core where it should go. The control tower asks all airports when they are ready for a new drone, and then chooses the airport which will be ready first. This airport is then returned to the drone, and the drone can start its approach for this airport.

The calculations done at each airport in order to find the next available service is based on how many drones are at the airport, how many drones are approaching the airport, and how far into a battery change it is.

```
1 Next available service = Num drones at airport * time to change a battery
2 Next available service -= progress on current battery change
3 For each drone approaching airport:
4     if drone arrival ≤ next available service:
5         next available service += time to change a battery
6     else:
7         next available service = drone arrival + time to change a battery
```

The pseudo code above shows how the airports calculate the remaining time until it is available. First, it adds up the time it takes to change the battery on the drones currently at the airport. The progress on the current change is then subtracted, since this is time the airport already has spent. Finally, the airport goes through all drones that are approaching the airport,

and calculate when the last of the approaching drones will be finished with their battery change. The operation of going through all approaching drones works as a recursive function. It starts with the one arriving first, and checks if this drone comes before or after the airport is finished with the previous one. If it arrives before the airport is ready, it means that the drone will have to wait and that the battery change for this drone begins as quickly as it finishes with the previous one. For the total time until the airport is available it means that it only needs to add the time it takes to change one battery to the total. The other scenario is that the drone arrives at the airport after the previous battery change is finished. This means that the drone can change the battery as soon as it arrives. For the total time it means that instead of simply adding the time for a battery change, it also needs to add the time where the airport waits for the drone. Then the airport can calculate for the second drone approaching, the same way. When the airport is finished going through all approaching drones, the total value is returned to the control tower.

## 3.7 Random methods

There are two interesting ways of looking at a random system. They are included since they seem to represent the simplest "optimizer" to implement, which shows how the system works with a pretty simple solution. The difference between the methods is whether the distance should affect the random selection or not.

### 3.7.1 Random

The basic idea of the random method is that the next airport is chosen completely random. This means that the airports, and/or drones, must know from the beginning which airports that are in the system. In the simulations done for this thesis, it is the airports that know where all other airports are. Therefore it is the airport that randomly chooses, with a uniformly spread, the next airport for the drone, and gives information about the new airport to the drone when changing its battery. Then the drone flies out and lands at the new airport next time it needs a new battery. This system simulates what happens when there is no correlation between the location of airports and where the drones operate, and therefore not possible to weight the randomness.

### 3.7.2 Distance weighted random selection

The second random method is some what smarter in the sence that it uses some of the infromation available to make a descision. The infromation used is the distance to the airports. If the distance to all airports are the same,

the random pick will be uniform as in the random method above (3.7.1). When the distance is different, the random pick will be weighted in the favor of the closest one meaning that it is more likely to choose the airports closest. In order to achive the correct distribution the method is implemented on each drone. This means that the drone must know where all airports are. When calculating the distribution, the drone creates an array containing how much time it loses by changing at the different airports. This means that all airports have the time it takes to change the battery plus the time it takes to travel to that airport. This array is then inverted so that the lowest value is the one furthest away. Finally, the array is divided by the sum of the array, to achieve the normalized array. The resulting array is then used as the weighted distribution for the random pick.

## 3.8 Response Threshold

When creating a swarm optimization method which is inspired by response threshold, there are mainly two interesting factors. Number one is what kind of information is distributed and how much, the second is how the agent figures out its reaction to this information. When it comes to what kind of information the drones should share, it is important to find the information that informs the most to other drones with as little information as possible. In a drone airport problem it seems natural to use information regarding how busy the airports are so the drones can use this to compare all the airports.

When one knows how busy the airports are, they can be given a value based on when they are available. This value represents how long time it is until the airport can change battery on the drone, and is used as stimuli for the drones. Since a low stimuli means that the airport is soon ready, while a large stimuli means that the airport is very busy, equation 2.2 is inverted compared to the decired probability. The equation is therefore updated in order to fit the decired inverted form, resulting in equation 3.5. By using equation 3.5 the probability of reacting is 1 when the stimuli is 0, while the probability converges towards 0 when the stimuli increases.

$$T_{\Theta}(S_i) = \frac{\Theta^n}{S_i^n + \Theta^n} \tag{3.5}$$

In equation 3.5 $S_i$ is the stimuli from airport $i$. For this method to work in a quickely changing environment, the response threshold $\Theta$ is implemented as a variable dependent on the different stimulis present. By setting $\Theta$ as the average over all stimulis present (as shown in equation 3.6), the method will always priorities the airports with lowest stimuli. In equation 3.6 $n$ is the total number of airports in system.

$$\Theta = \frac{\sum_{i=1}^{n} S_i}{n} \tag{3.6}$$

### 3.8.1 Implementation

Like the random methods, the only difference between the two swarm methods is how much the drones know of the initial state. The first one knows about all the airports in the system when it starts, and can, therefore, choose to go to all airports from the start. The second method does not know about any other airports from the start, and can therefore only chose to go back to the one it came from, or it must learn about other airports from other drones. Communication is crucial to get more information about the system. Communication is done through the grid explained in section 3.3.7.

As explained above, it is important to find the information which is the most informative with as little information as possible. The information chosen in these simulations are:

- Drone ID
- Previous airport
- Location of airport
- Number of drones at the airport when departed
- Departure time from the airport

The first item on the list is most important in order to keep track of the newest information from each drone. Information gets old really quickly and there is no reason to keep track of all the old information. This means that every time a drone receives information, it checks if this information has already been received. If so, it just throws it away. If it is new information, the drone keeps it. If it has met this drone earlier, it replaces the old information.

The next four items on the list informs the other drones about the activity at the previous airport the agent had service. First of all, the information informs which airport it concerns, and where to find it. The location of the airport is used to tell the other drones how far away it is. The final two items informs the other drones how busy the airport was when the agent left. This information tells a lot to the other drones when the information is new, but if it's old it might not tell anything at all.

When knowing which information is available, the method explained above can be used. For the simulations, a greedy version of the method has been used. This means that the drones always choose the best solution based on the stimuli received. This is done as in equation 3.7 where $T_\Theta$ is calculated by equation 3.5.

$$\text{next airport: } A = \max(T_\Theta(S_i) : i = 1, .., n) \tag{3.7}$$

How this process is implemented can be viewed below.

```
1 For each airport heard of from other drones:
2    probability = calculateProb(airport)
3    if probability > bestProbability:
4        save new best probability & new best airport
```

## 3.9 Collecting data

In order to compare the methods after running the simulations, one must collect data from the simulations. One simulation contains several runs with the same method. One run contains one "lifetime". This means that one run consists of creating x number of airports and y number of drones, and running them for z number of clicks. After z number of clicks, all agents terminate, and the run is over. For each run, the program calculates the average number of drones in the air and saves this value. The program also saves how many battery changes each of the x number of airports perform. This way it is possible to see which of the airports are the most active. For most of the simulations, each run is done several times in order to achieve an average for both the number of drones in the air and the number of battery changes per airport. By creating an average over several runs with the same configuration, the result is a better representation of this specific configuration.

# Chapter 4

# Simulation experiments

In this chapter, all experiments will be explained fully, from the specific setup for the different configurations, to the results and analysis. The first subsection will explain assumptions and choice of parameter values in general, while more details specific for the different configurations, will be presented at the beginning of their own sections.

## 4.1 Simulation setup

In order to go from the implementation explained in the previous chapter, to actually running the experiments, there are several choices to make regarding parameter values. Table 4.1 sums up the key parameters, and which values that have been used in all experiments.

| **Parameter** | **Value** |
|---|---|
| Size of drone work space | 900 X 500 |
| Drone speed | 1 pixel/click |
| Battery loss | 0.05 %/click |
| Battery after change | 80-100 % |
| Return percentage | 15% |
| $T_{bc}$: Number of clicks it takes to change battery | 300 |

Table 4.1: Summary of the parameter values choosen for the experiments.

All of the parameters listed in 4.1 affect mostly the drones. In some way, they all decide how long the drone can stay in each of its 3 states mentioned in table 3.1. The size of the workspace affects how far away from the airport a drone can fly. Since the targets a drone has are random throughout the whole workspace, an increasement of the workspace also increases the average approach time for the drones. The speed, on the other hand, is the opposite. The more it increases the lower the average approach time, as it can travel faster.

The three next parameters affect how long the drone stays in the air, while the final one effects how long the drones need to be in its landed state. When it comes to the airport, the four last parameters affect the saturation point, as explained for equation 3.1.

## 4.2 Results with one airport

The first results shown will answer how well it is possible to optimize one airport. This is done using the mathematical optimization method. This will also show how many drones will saturate an airport.

### 4.2.1 Setup

When simulating one airport, it was important that the configuration was the same as for several airports, except form the fact that there is only one airport. The placement of the airport is in the middle of the workspace, which gives the shortest average traveling distance for the drones. To be sure that the airport is fully saturated, the simulations went from 0 to 14 drones in the swarm. To be sure that the results are good representations, each configuration of drones has been run 10 times. The average of the 10 runs is what is plotted together with the standard deviation. Based on the standard deviation it seems that 10 runs is an adaquate number of runs to represent the results.

To get an understanding of how much the mathematical optimization actually optimizes, there is also included a random method in the results. This method only returns the drone to the airport when it needs to change the battery.

### 4.2.2 Results

Figure 4.1 shows the average percentage of the swarm which is active. A drone is active when it is on a "mission", meaning not at an airport, or on its way to an airport. Figure 4.2 shows the average number of active drones in the swarm. From figure 4.1 we can see that when the system is under-saturated, about 70% of the swarm is active for both optimization methods. There is a small decrease in the percentage before the saturation point. For the mathematically optimized method (blue line) this is most likely a result of the drones having to either wait because there is another drone at the airport, or land a little early in order to be done when the next arrives. For the random method it can only be that the drones are waiting, since they will never arrive before they have to. Since there isn't any sudden decrease, before the saturation point it seems like the drones don't have to wait for long.

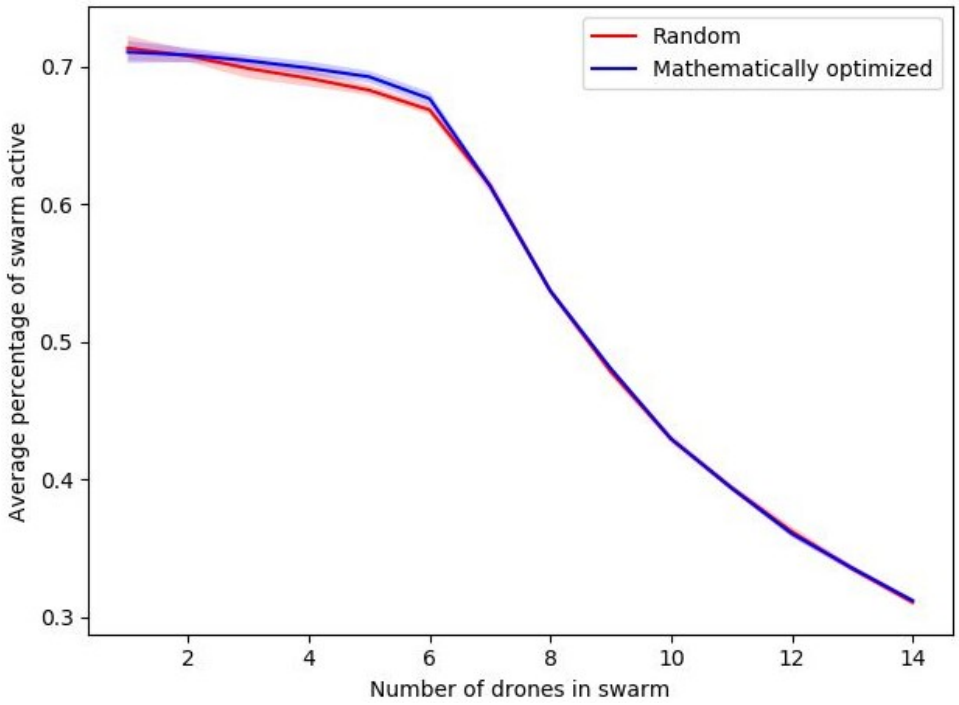


Figure 4.1: Average percentage of drones active with one airport.

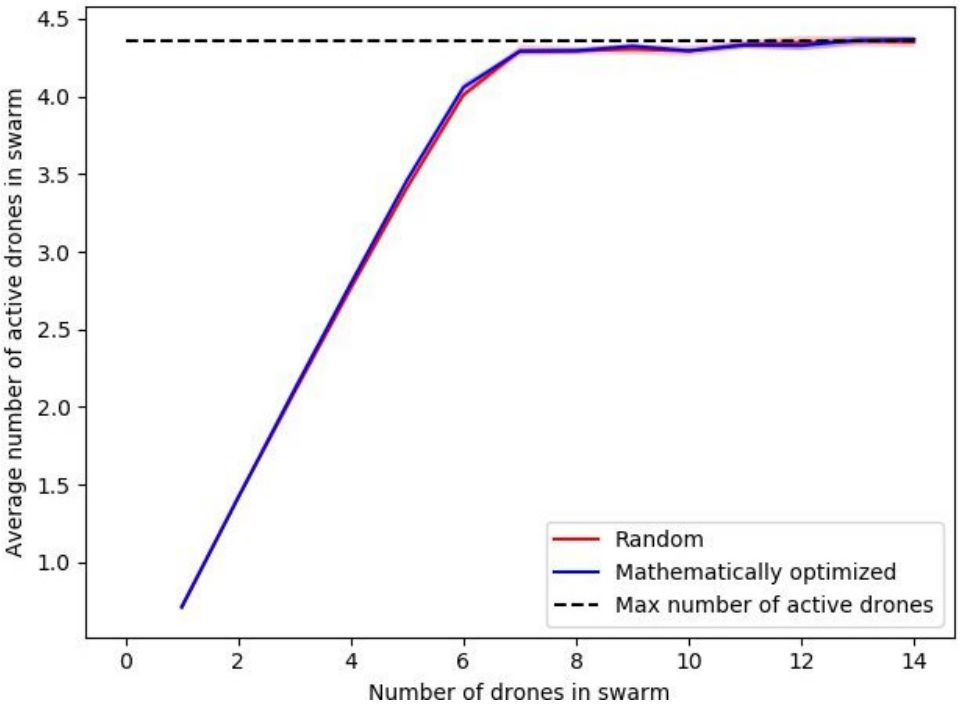


Figure 4.2: Average number of drones active with one airports.

Moving on to the saturation point it seems to be between 5 and 7 drones for both methods. By looking at figure 4.2 it seems that the lines flattens out when it reaches 7 drones. Interestingly, the change between 6 and 7 drones is smaller than between 5 and 6, meaning that the saturation point is somewhere between 6 and 7 drones. Equation 3.1 from section 3.3.5, was the equation for calculating the saturation point of an airport. When entering the configuration of the simulation into this equation, the resulting saturation point is 6. Since the battery a drone receives when changing battery is between 80% and 100%, it is assumed in the calculation that the average battery percent level when departing the airport is 90%. This means that the experiments and the calculation is rather synchronized.

When moving past the saturation point to the over-saturated state, figure 4.2 shows that the average number of active drones converge towards approximately 4.35.

In general there is very little difference between the random method and the mathematically optimized. In figure 4.1 we can see that it is only in the area between 2 and 6 drones that the random method lies a little bellow the mathematical method. This shows that there is very little to gain by optimizing only the slot as done in the mathematical method.

## 4.3 Results with several airports

### 4.3.1 Setup

The results that will be viewed and analyzed in the next sections are collected from several experiments. Table 4.2 gives a quick overview of all experiments done. As described in previous chapters, the methods of interest are centralized, 2 random methods and 2 swarm methods. Since the centralized method is dependent on full communication, it is only simulated in this state. Also, the random methods have no communication and are therefore only simulated with limited communication, since lack of communication is a form of limited communication.

The experiments group up in two main groups. One for experiments where the airports are spread, and one where they are at the same location. Each group of experiments can again be divided into two groups. The two undergroups regard the communication range. In table 4.2, full communication means that all drones can communicate with all other drones wherever they are. Limited communication means that the drone workspace is divided into 180 equal sized squares which form the grid explained in the implementation chapter. Mark that the second random method (R2) is not simulated in the configuration where the airports are at the same place. This is because in this configuration the random selection is equal to R1.

Within the two undergroups of experiments, lies all methods each represented as one experiment. When running one experiment, all configu-

ration of parameters stays the same except the number of drones in the system. Each experiment contains 600 runs where the number of drones varies from 1-60 drones. This range of drones is choosen since it goes from the most under-saturated stat possible, until the system should be really over-saturated. Each number of drones is run 10 times before collecting the average number of drones in air and number of battery changes per airport, as explained in section 3.9. As can be viewed in the result plots later, the standard deviation is very smal confirming that 10 runs is a sufficiant amount of runs per configuration. One run is defined as a "life cycle" for the drones. When starting a new run, all agents are created and simulate their movements for 100 000 clicks. Each click the drones and airports act according to the implementation explained in an earlier chapter. All the parameters above are also choosen as a result from how long time it takes to run the experiments. In order to run the experiments with the parameters as above it took 2 weeks of cpu-time. Had one for example increased the number of clicks to 1 million, the cpu-time would have been increased to about 20 weeks. This time was unfortunatly not available.

In table 4.2 each experiment i given a code corresponding to what it is. As an example, AC-FC-C1 means that the airports are centralized, full communication, and the centralized control method. This code will be used in all plots when referring to the different methods.

| | | |
|---|---|---|
| Airports centralized (AC) | (FC) Full communication | Centralized (C1) |
| | | Swarm with initial information (S1) |
| | | Swarm without initial information (S2) |
| | Limited communication (LC) | Random (R1) |
| | | Swarm with initial information (S1) |
| | | Swarm without initial information (S2) |
| Airports spread out (AS) | (FC) Full communication | Centralized (C1) |
| | | Swarm with initial information (S1) |
| | | Swarm without initial information (S2) |
| | Limited communication (LC) | Random (R1) |
| | | Distance weighted random selection (R2) |
| | | Swarm with initial information (S1) |
| | | Swarm without initial information (S2) |

Table 4.2: Summary of the different experiments.

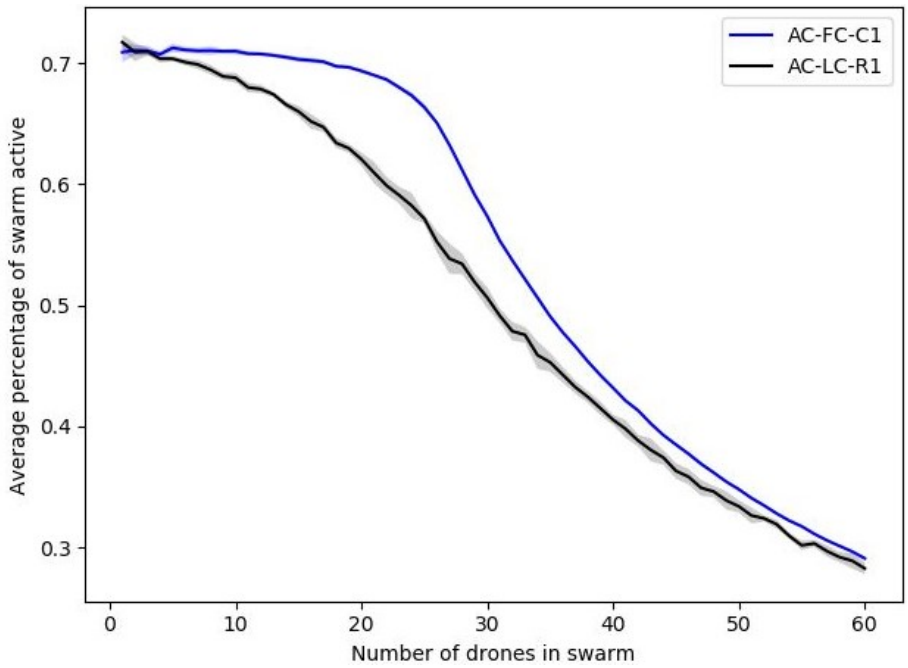


Figure 4.3: Average percentage of drones active: Benchmark methods.

## 4.4 All airports at same place

### 4.4.1 Benchmark methods

The benchmark systems will give a general understanding of how well the system works before going in depth with in the swarm methods.

#### Results

Figure 4.3 and 4.4 shows how well the benchmark systems work. Figure 4.4 shows the average number of drones that are active at any given time. When a drone is active, it is not at an airport or on the way to an airport. Figure 4.3 displays the same data, but shows how many percent of the drone swarm is active. In both figures, the blue line represents the centralized method, and the black line represent the random method. The dashed line in figure 4.4 shows the maximum number of active drones that the system converges towards.

Figure 4.5 and 4.6 shows how the system divides the drones between the airports. Figure 4.5 shows how the centralized method devides the load between the airports, and figure 4.6 show the same for random optimization.

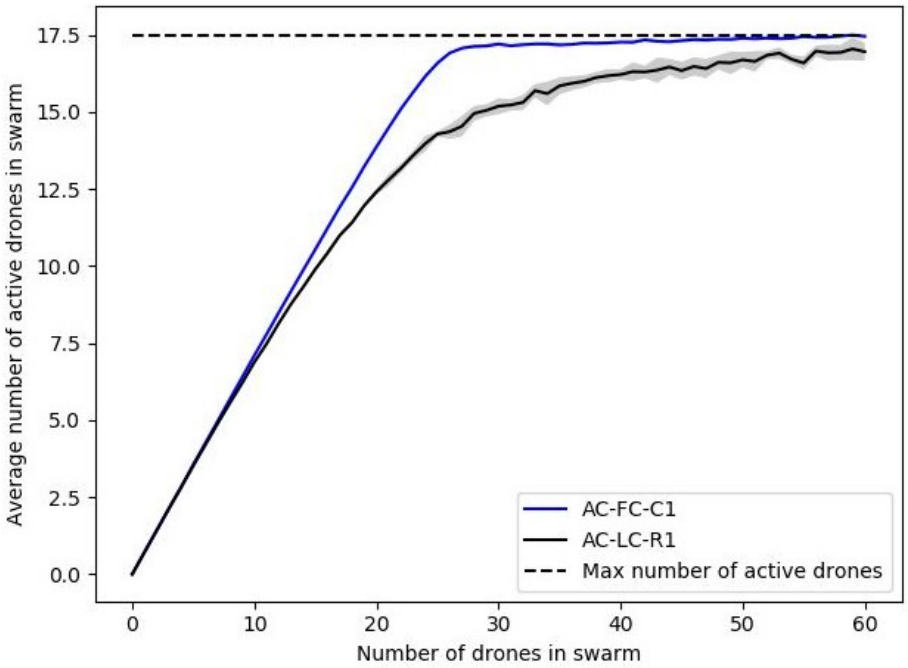


Figure 4.4: Average number of drones active: Benchmark methods.

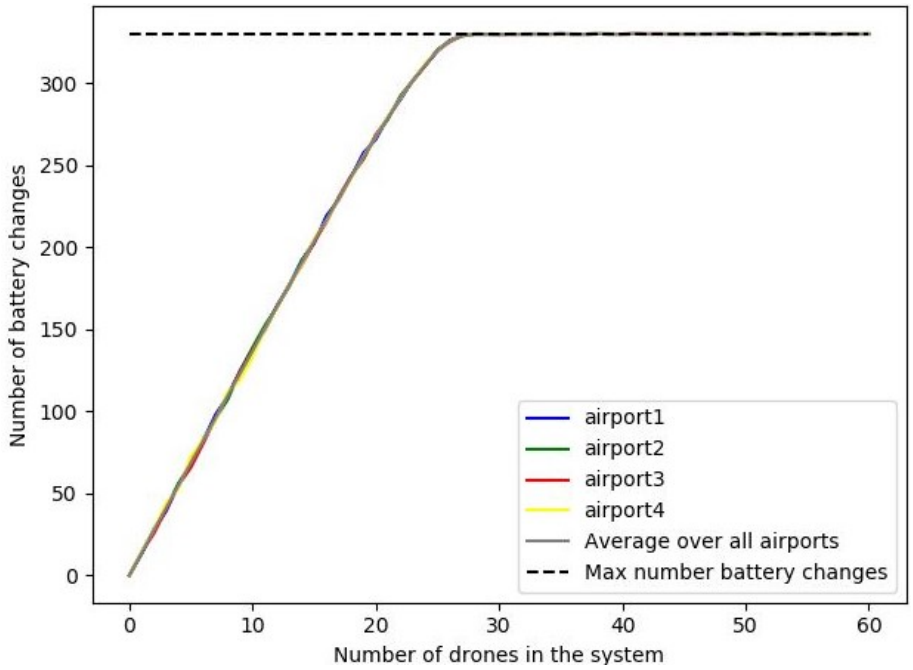


Figure 4.5: Number of battery changes per airport. Centralized optimization, all airports at the same place.

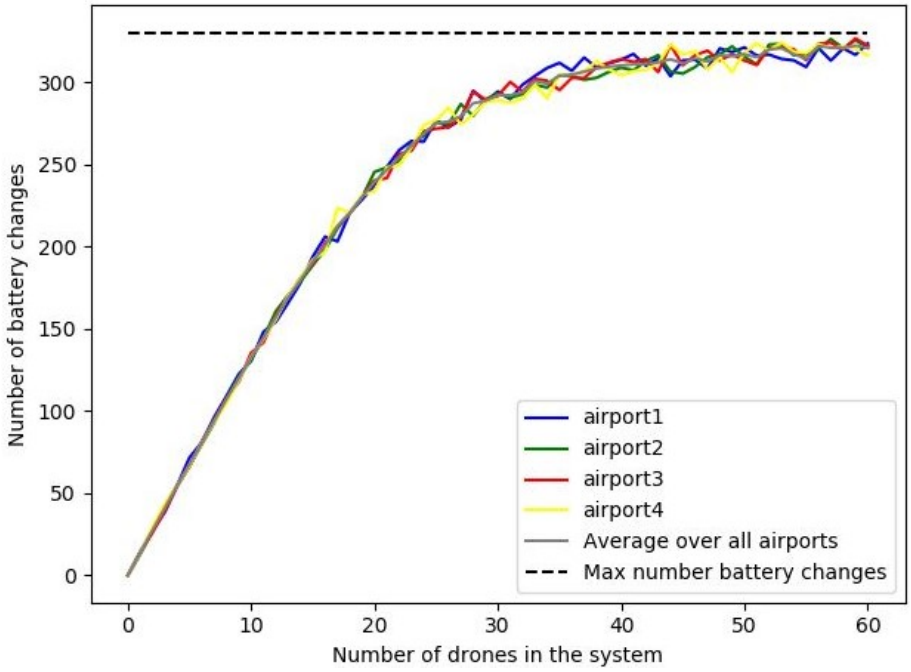


Figure 4.6: Number of battery changes per airport. Random optimization, all airports at the same place.

### Analysis

By looking at figure 4.4, the centralized method has almost a linear increas from 1 to 20 drones. In this area, the system is not yet saturated, and therefore the system can handle additional drones. This is confirmed by looking at figure 4.3 where the centralized method is almost linear and stable at 70 percent between 1 and 20 drones. From the figures one can also see that the saturation point is between 20 and 30 drones when both figures show the transition from under-saturated to over-saturated. Especially in figure 4.4, one can see that somewhere between 20 and 30 drones the system is no longer able to increase the number of drones active. In fact, the number of active drones only increases from 17.2 to 17.5 between 30 and 60 drones. This means that the system converges towards 17.5 drones active which is marked by the dashed black line. From the experiment with mathematical optimization on one airport, the results were 4.35 active drones in the system. Multiplying this with four airports results in 17.4 which corresponds quite nicely with the results from several airports.

It is more difficult to find a specific point where the random method enter a saturated state since the transition between under-saturated and over-saturated is smoother. Even though the random method does not reach the 17.5 mark in figure 4.4, one can see that it also converge towards it.

Concerning how the system divides the load, it seems that the system successfully divides the load on all airports from the beginning when using centralized optimization. This results in a quite linear increas for all drones, as can be seen in figure 4.5. When looking at the random method, figure 4.6 shows that it devides the load quite similar as the centralized method in the beginning. When more drones enter the system it seems like it still manages to divide relativly equal to all airports, but as we saw in figur 4.4 there is no clear saturation point. Nontheless, figure 4.6 show that the random method seem to have a uniform distribution.

Figures 4.5 and 4.6 also show the same form as the corresponding methods in figure 4.4. This confirms the relationship between the number of battery changes in total and the number of active drones in the swarm. The more battery changes done by the airports, the more drones stay active in the swarm. It is also interesting to see that the centralized method quickly converges towards the max number of battery changes per airport, while the random method does not entirely reach it at any point. The maximum possible battery changes for any airport is given by the total number of clicks simulated divided by the number of clicks it takes to change a battery: $100000/300 \approx 333$. The maximum number of battery changes actually performed in any of these experiments is 331. This is due to the initial battery state on the drones when the experiments start. This is set to a random percentage between 50 and 100%. Since none of the drones need battery change before their battery percent reaches 15%, it will take at least $\frac{(50-15)\%}{0.05\%/click} = 700$ clicks before any drone needs a battery change. Taking this into account, the maximum possible number of battery changes for an airport is $(100000 - 700)/300 = 331$, which confirms what the experiments sometimes experience.

### 4.4.2 Swarm methods

#### Results

Figure 4.7 and 4.8 shows how well the swarm methods performed. The four swarm methods represented vary in terms of communication range and how much they know about the initial state. As explained in chapter 3.8.1 the amount of information obtained from the start effects which airports the drone can choose amongst. Figure 4.9 and 4.10 show the division of labor between the airports for swarm optimization with limited communication and swarm optimization with full communication, respectively.

#### Analysis

By looking at the figures there seems to be no effect on the system whether the drone initially knows where all airports are or not. The communication range, on the other hand, seems to affect the system. The swarm methods

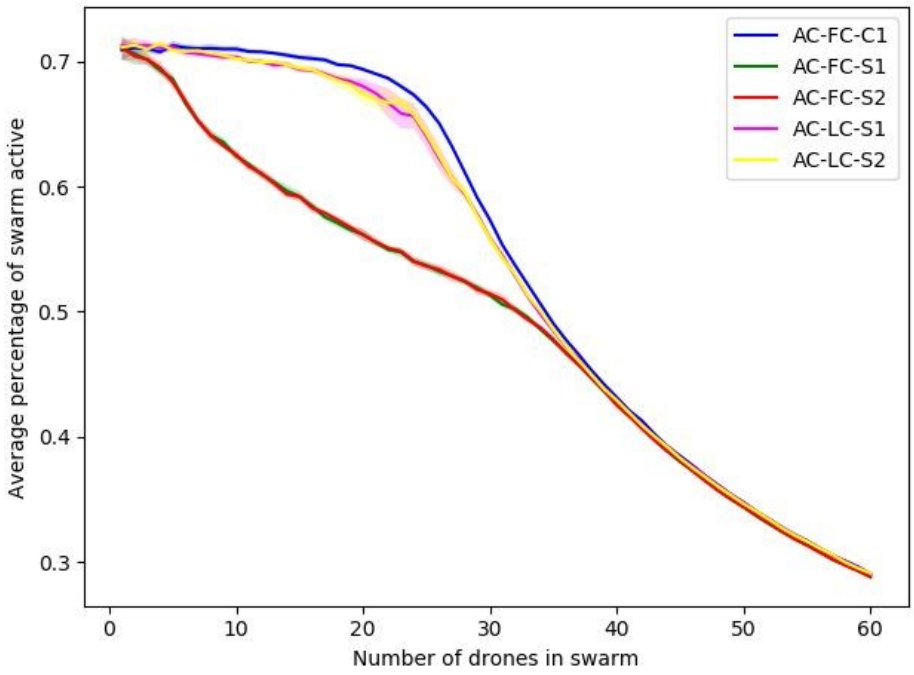


Figure 4.7: Average percentage of drones active.

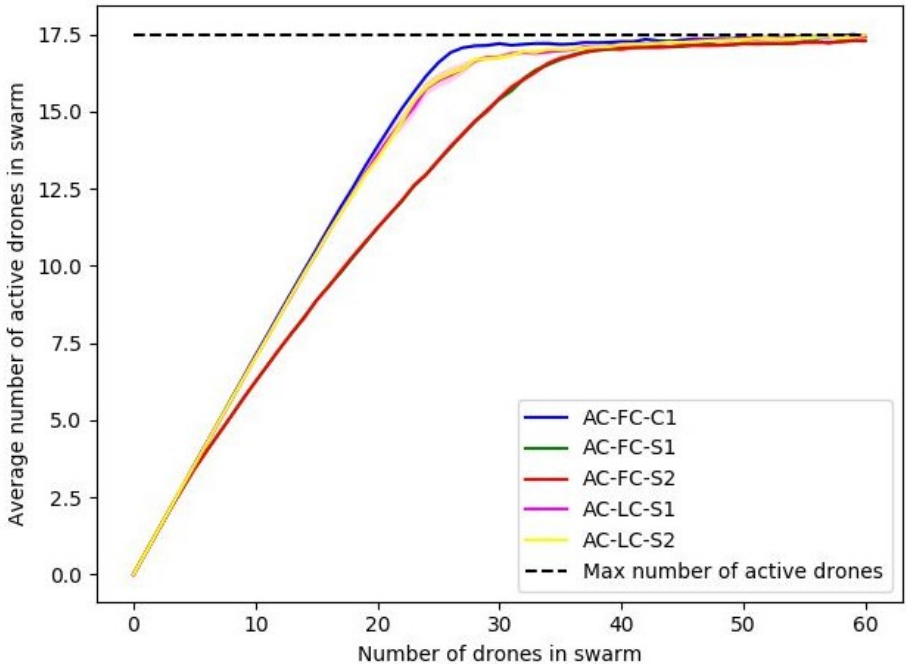


Figure 4.8: Average number of drones active.

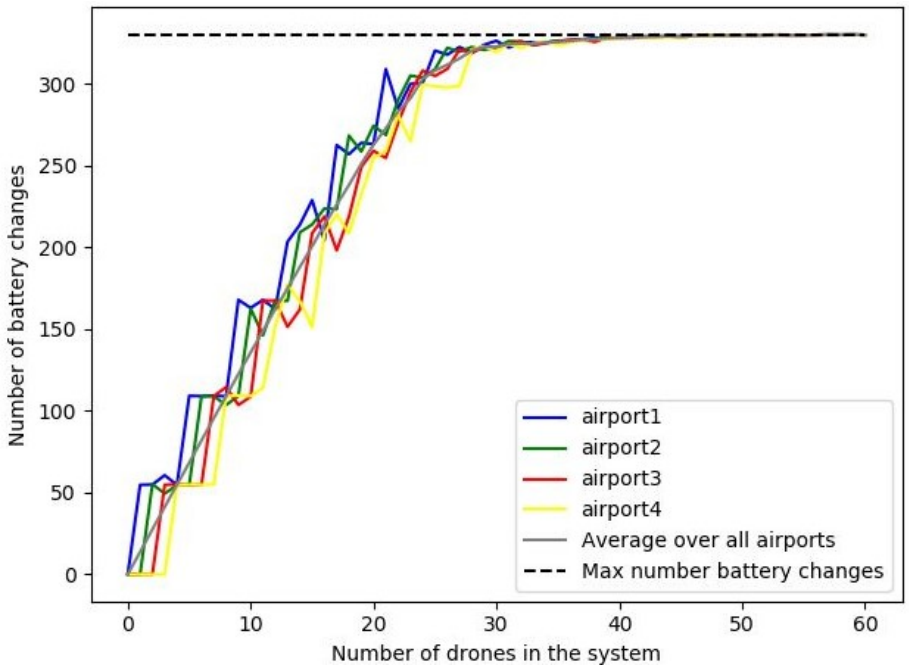


Figure 4.9: Number of battery changes per airport. Swarm optimization, all airports at the same place and limited communication.

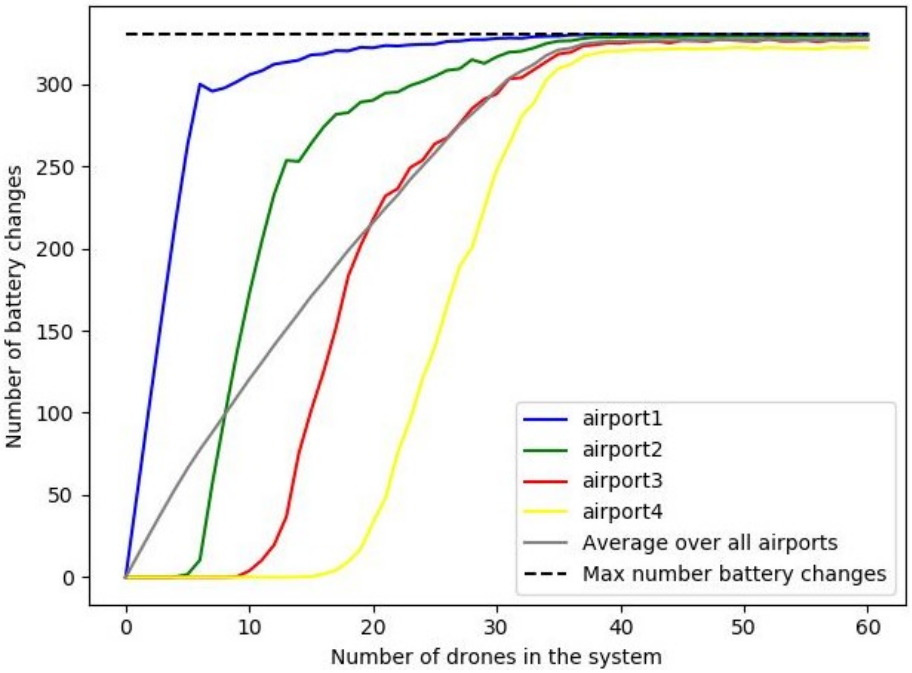


Figure 4.10: Number of battery changes per airport. Swarm optimization, all airports at the same place and full communication.

that have limited communication seem to perform better than the method which uses full communication. One possible reason for this is that with full communication all drones have access to the same information, and will therefore possibly make the same decision regarding airports without knowing that someone else also selected the same one. This can result in several drones flying towards the same airport at once, and therefore need to wait for a long time. In the method with limited communication however, the drones have different access to information which seems to results in a more spread distribution.

The experiments where communication was limited (yellow and purple line) seems to follow a similar form as the centralized communication, where the increas of the system is quite linear between 1 and 20 drones. The saturation point appears to come between 20 and 30 drones before it flattens out in figure 4.8 and converges towards the same 17.5 mark. When it comes to the experiments with full communication (red and green line), there is a more smooth transition surrounding the saturation point. Looking at figure 4.7 the average percentage seems to have an almost constant loss when increasing the number of drones.

By looking at figure 4.10 it seems that the constant loss experienced with full communication is a result of the division of labor between the airports. For some reason, the drones are first only using one airport, until they experience some sort of queue at this airport, at which time the next one is also used. One explanation for this can be an implementation detail concerning arrays and how the communication grid works. First of all, when a drone is created, it is also put into a list for the communication grid square it is in. Since drone 1 is always the first drone created, it is also the first drone put into this list. Drone 1 is always assigned to airport 1. When there is full communication in the system, all drones are put into the same list since the whole workspace is in the same communication square. This results in drone 1 always being the first entry to the list. After all drones are assigned to the list, they start going through it to find new information. The first new information it finds, which is always drone 1 the first time it checks, will be the first airport added in the drones overview. When the drone then needs to find which airport it will go to for its battery change, it will always check airport 1 first. After checking airport 1, the other airports will have to be estimated as a better solution than number 1 in order to take its place as the best solution. If the estimation is equal, the drone will keep the first one as its best solution. Not only is airport 1 always the first to be checked, but airport 2 is always number two, since drone 2 is the second to be created and therefore the second drone assigned to the communication list. Likewise is airport 3 the third, and airport 4 the fourth.

The problem regarding the order of which the airports are checked, explains the spread in regards to the airports in figure 4.10, but it does not alone explain why the results in figure 4.7 and 4.8 are so much worse than for

limited communication. In order to fully understand the different results, one must also understand what is happening throughout the simulation, and especially how the information is spread. When a drone changes its battery, it is removed from the communication list mentioned above. When it is finished it is again assigned to the list with new information regarding its latest battery change. The information communicated from the drone does not change until the next time it changes its battery. Since this happens for all drones, the information regarding an airport is only updated every time the airport finishes a battery change. Since a battery change takes 300 clicks in these experiments, it will take at least 300 clicks until new information about the airport is available. This means that the information can get rather "old" by the time the airport has changed the battery on a new drone. By the time new information enters, several drones can have made a choice for their next airport, which can potentially have led many of them to choose the same airport. This effect can happen for all airports, and can result in sending batches of drones to each airport, resulting in many drones having to wait in line.

Analyzing figure 4.9, one can see that with limited communication the same effect occurs, but is not as dominant since the drones cannot communicate at the beginning. When the drones start, they are spread out over the workspace and located in different communication grid squares. This means that they are not listed in the same communication list, and therefore do not get the same airport as the first one they are informed about. In figure 4.9, there is almost a stairs phenomenon the first couple of drones. This shows that when a new drone is included, it will mainly use its origin airport when changing the battery. It is the same effect explained regarding figure 4.10, that the drone usually returns to the first airport it hears about. We can also see in figure 4.9 that when there are enough drones in the system, this effect slowly decreases, and each airport quickly converges to the maximum number of battery changes.

### 4.4.3 Comparing methods

In order to fully understand how well the methods are optimizing, they must be compared to one another. In figure 4.11, all methods are represented and they are color separated. The centralized method is the blue and the random method is in yellow. As for the swarm method, they are divided into two colors. Green for limited communication, and red for full communication. As a first remark, the centralized method is, as expected, optimizing the best. Closest to the centralized, and a very promising result, is the swarm method with limited communication. It has almost the same performance as the centralized method and shows that it can be possible to optimize very well even without full communication or centralized control. An interesting observation is that the system performs just as well regardless of if the drones

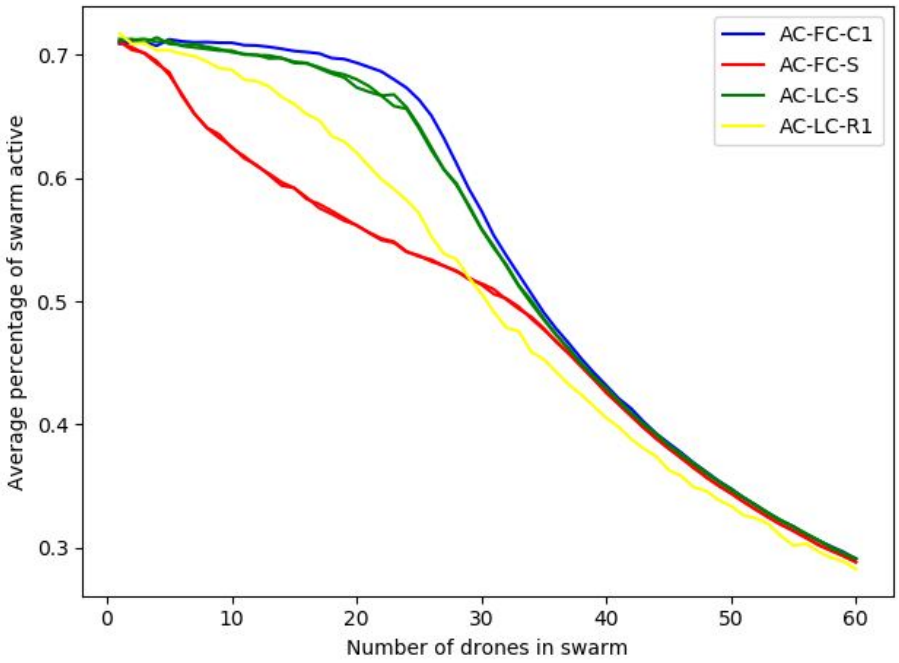


Figure 4.11: Average percentage of drones active: All methods.

know about all the other airports from the beginning or not. This means that communication must work pretty well, and the lack of communication does not affect the system in a bad way.

When it comes to the random method, the results are actually surprisingly good. It shows that the system can work rather well without any need for logical/smart drones. The negative part about the random method is that it needs a lot of time to converge towards the maximum number of drones active as shown in figure 4.4.

The not so promising results concern the swarm method with full communication. As explained in the analysis earlier the problem with all drones always prioritizing the same airport seem to affect the optimization badly. This problem could be fixed by changing the implementation of how the drone updates its overview information, or by changing the method used to pick airport. The positive part about this method is how quickly it converges when the number of drones reaches an over-saturated state. Another possible implementation of the swarm method when there is full communication could be to make it more like the centralized method. Since there is full communication in the swarm, each drone can do the same calculations as done in the centralized unit, only for themselves. This would mean that the airports must also be connected to the communication grid, which they are not in the simulated configurations.

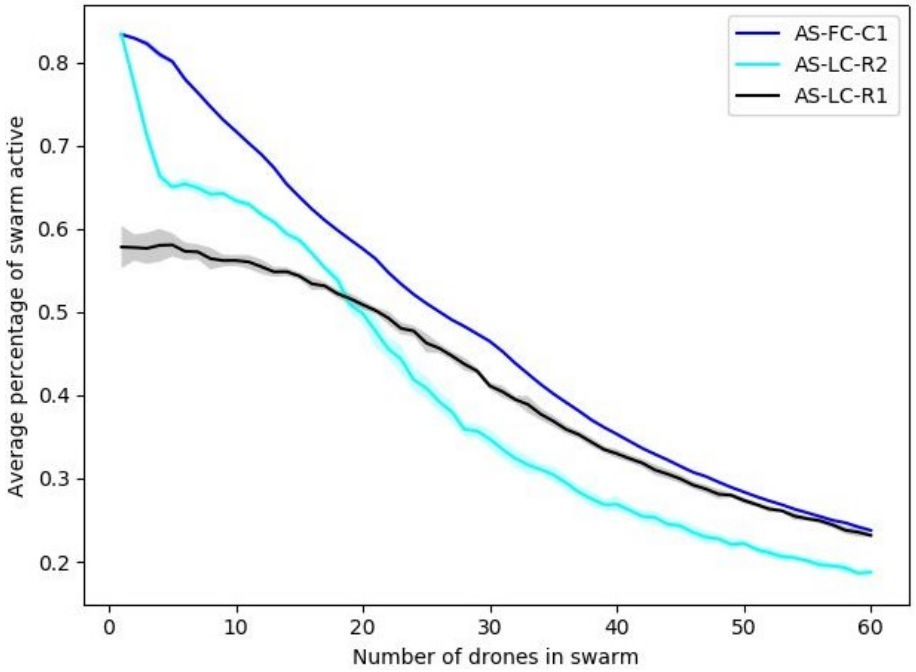


Figure 4.12: Average percentage of drones active: Benchmark methods.

## 4.5 Airports spread

### 4.5.1 Benchmark methods

#### Results

When looking at the configuration where the airports are spread, the results look a little different. Figure 4.12 and 4.13 show how the benchmark methods performed. Again the blue line represents the centralized method, while the cyan and black lines are the different random methods.

Figure 4.14, 4.15 and 4.16 show how the different methods divides the battery changes between the airports.

#### Analysis

The centralized method performs quite well in this configuration as well. It does not have the same linearity as when all airports had the same location, but it quickly achieves a relatively high average of drones that are active. The system clearly reaches a point where it seems to flatten out, but it needs some time to fully converge towards the maximum number of active drones (as seen in figure 4.13). Another interesting fact is that the maximum number of active drones is lower than in the last section. This is most likely a result of an averagely longer traveling distance to the airports. In figure

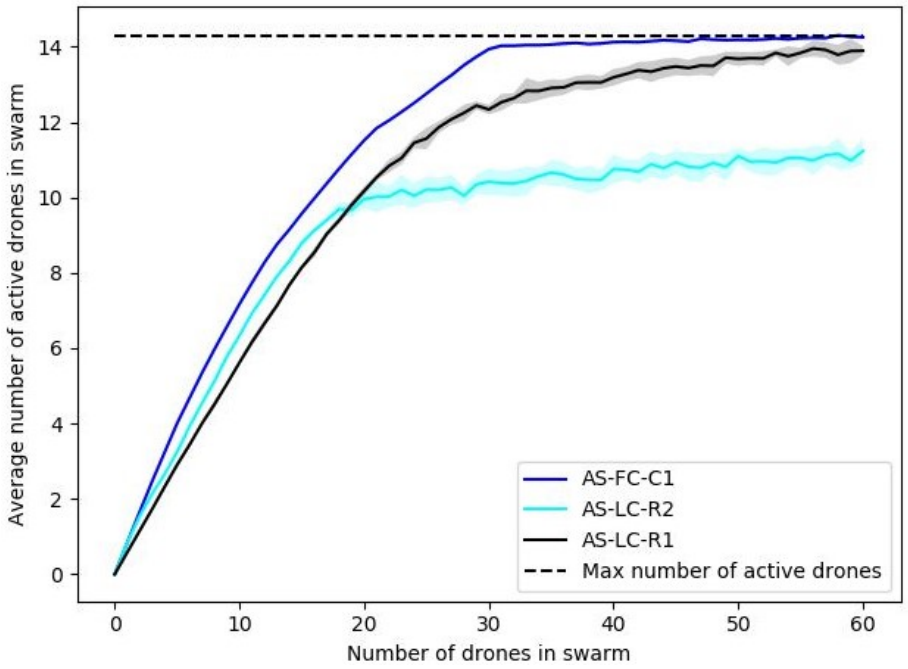


Figure 4.13: Average number of drones active: Benchmark methods.

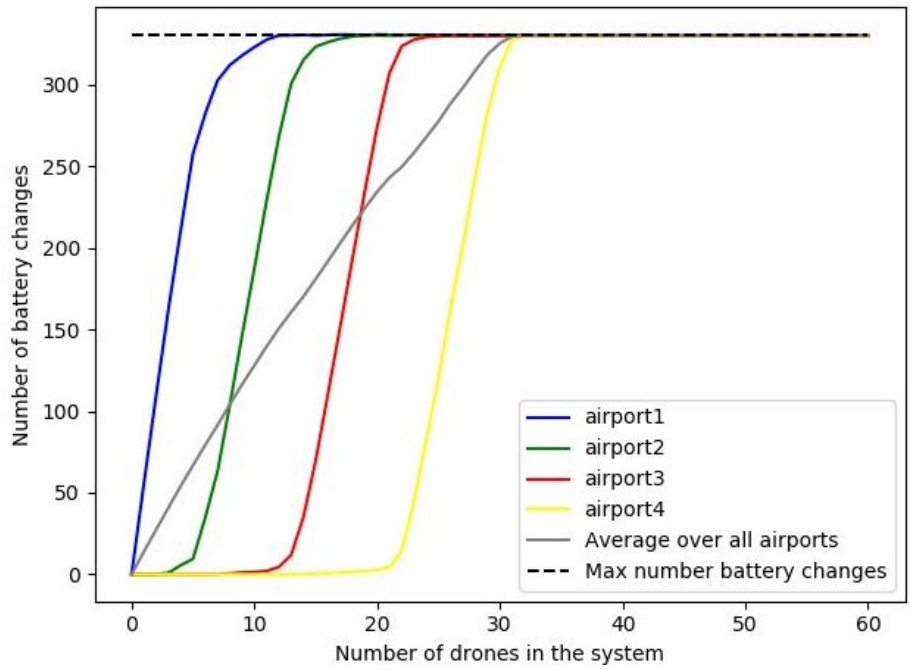


Figure 4.14: Number of battery changes per airport. Centralized optimization.

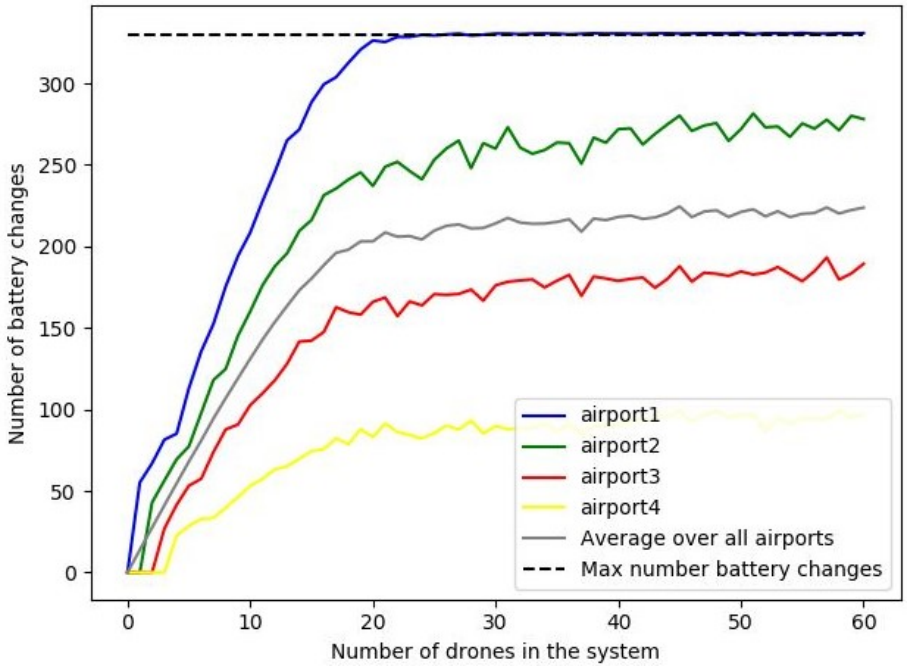


Figure 4.15: Number of battery changes per airport. Random optimization with distance weighted selection.

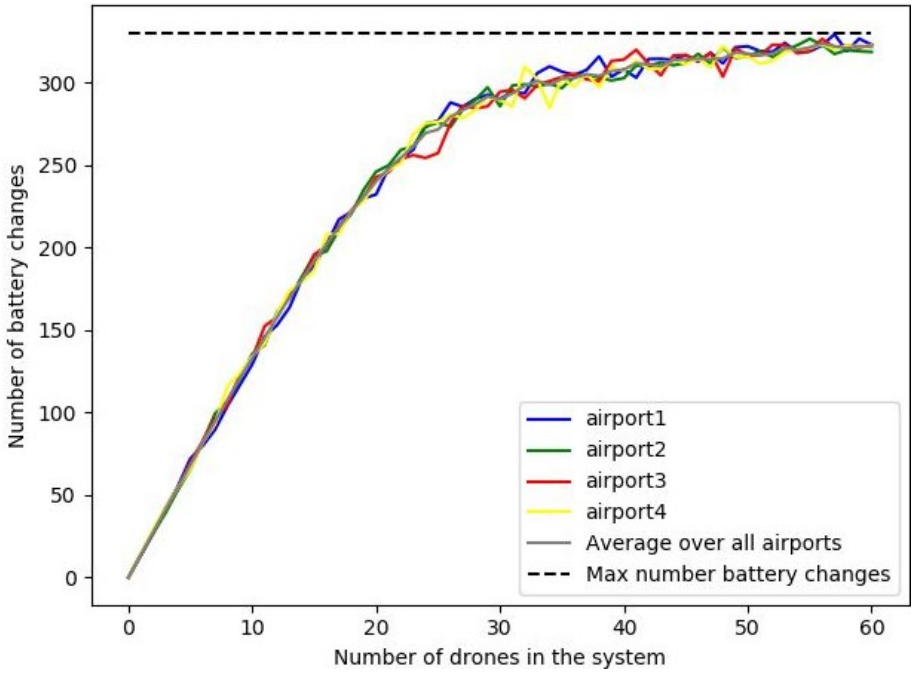


Figure 4.16: Number of battery changes per airport. Random optimization.

4.12 one can also see that the centralized method has a more constant loss to the percentage throughout the simulation compared to the configuration in the last section.

Figure 4.14 show how well the centralized method divides the drones in between the airports. It seems that each airport is almost filled to its saturation point before a new airport is being used. The explanation for this is a result of the different distances. If a drone goes to airport 2 instead of airport 1, it will need to use the equivalent of one extra battery change of time just in transportation. This means that the waiting time at airport 1 must exceed a whole battery change before it is beneficial to go to airport 2. By looking at figure 4.14 it seems that the waiting time can exceed a whole battery change when there are about 4-5 drones in the swarm since this is when the second airport is starting to be used. The third starts being used around 9-11 drones, while the fourth is not used much before 20 drones are in the system. In figure 4.13 one can actually see a sudden change of direction for the blue line around 20 drones, which is believed to be when the fourth airport is starting to be used.

For the random method where the distance to the airport is being accounted for (cyan line in figure 4.12), the performance starts out pretty well, but quickly decends. This is most likely since it is beneficial to prioritize airport 1 in the beginning. One can see in figure 4.13 that the method works best when there are less than 20 drones in the system. This shows that the weighting of the random pick based on distance is beneficial when the system is under-saturated and does not need to use all airports full time. When the system is over-saturated, one can see from the plots that the weighted selection is limiting the system, and therefore making it worse.

In figure 4.15, one can see that the weighting works as expected. Due to the different distances to the airports, the distribution between the airports is:

| Airport | Weight |
|---|---|
| Airport 1 | 40% |
| Airport 2 | 30% |
| Airport 3 | 20% |
| Airport 4 | 10% |

This means that airport 1 on average receives 4 times more drones than airport 4. As mentioned, this is good when there are few drones in the system, but when the amount increases all airports other than number 1 are not able to fully utilize their capacity.

The other random function where the airports are weighted equally (black line in 4.12 and 4.13), the results are opposite. It starts out pretty bad, only keeping a relatively small percentage of the swarm active. When increasing the number of drones in the swarm the percentage loss is rather

low. The transition between the under-saturated system and the over-saturated system is quite smooth with no sudden change in figure 4.13. The method never fully converges towards the maximum number of active drones, but is very close when there are 60 drones in the system.

Looking at figure 4.16 it is confirmed that the method works as it should. There is clearly a random selection done, and all airports are used equally. When the system is over-saturated, all airports almost converge towards the maximum number of battery changes possible which again explains why the black line in figure 4.13 almost converges towards the maximum number of active drones.

### 4.5.2 Swarm methods

#### Results

Figure 4.17 and 4.18 show the results of the swarm methods implemented in the configuration where airports are spread out with different distances. As in previous sections, one figure shows how large percentage of the swarm is active on average (4.17), and one shows the actual number of active drones (4.18). There are four different swarm methods tested and displayed in the figures. Two of the swarm methods are with limited communication, where the difference is if they know about all airports from the beginning or not. The same applies to the two methods where the drones have full communication.

Figures 4.19 and 4.20 show the number of battery changes per airport. 4.19 represents the results for limited communicaiton and 4.20 represents full communicaiton since the amount of initial information did not give any noticable effect on the plots.

#### Analysis

By looking at figures 4.17 and 4.18 it is clear that all the swarm methods perform quite similar. As mention above, it seems that the amount of initial information does not affect the system. When there is full communication, all drones immediately communicate with all drones and therefore know about all the airports. When there is limited communication there is a visual difference in the figures, but the difference is so small that it is negligible. This also shows that even though the communication is rather limited, the drones are able to spread the information pretty well. The difference between full and limited communication is also very small, but in general full communication lies a little bit above limited.

The methods seem to perform rather well the first 6-7 drones until there is a sudden change of direction for all swarm methods. In figure 4.19 and 4.20 one can see that airport 1 is filled up perfectly between 0 and 6 drones, before airport 2 is being used. Going back to figure 4.18, it can seem like

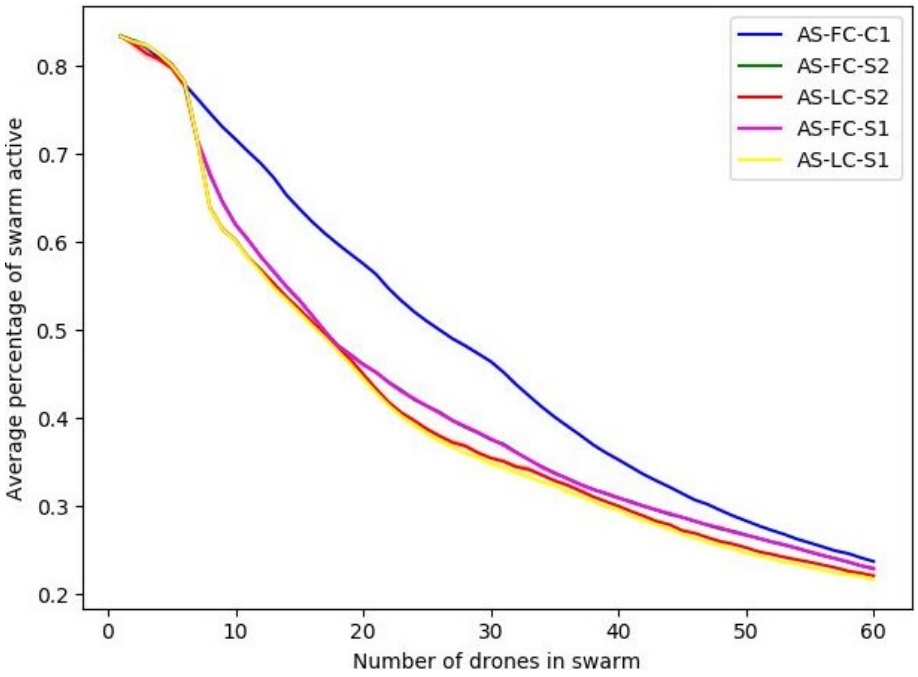


Figure 4.17: Average percentage of drones active.

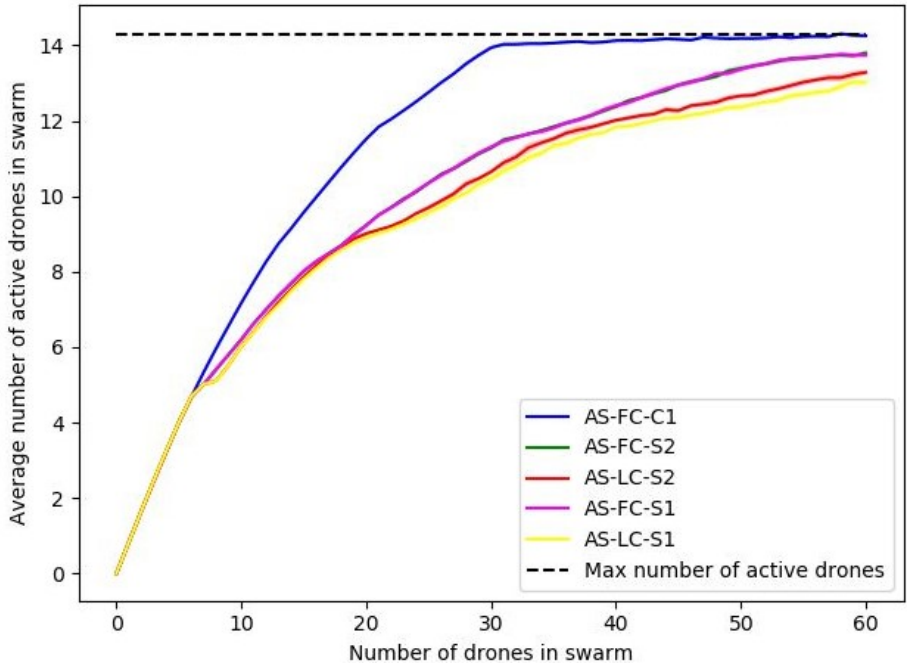


Figure 4.18: Average number of drones active.

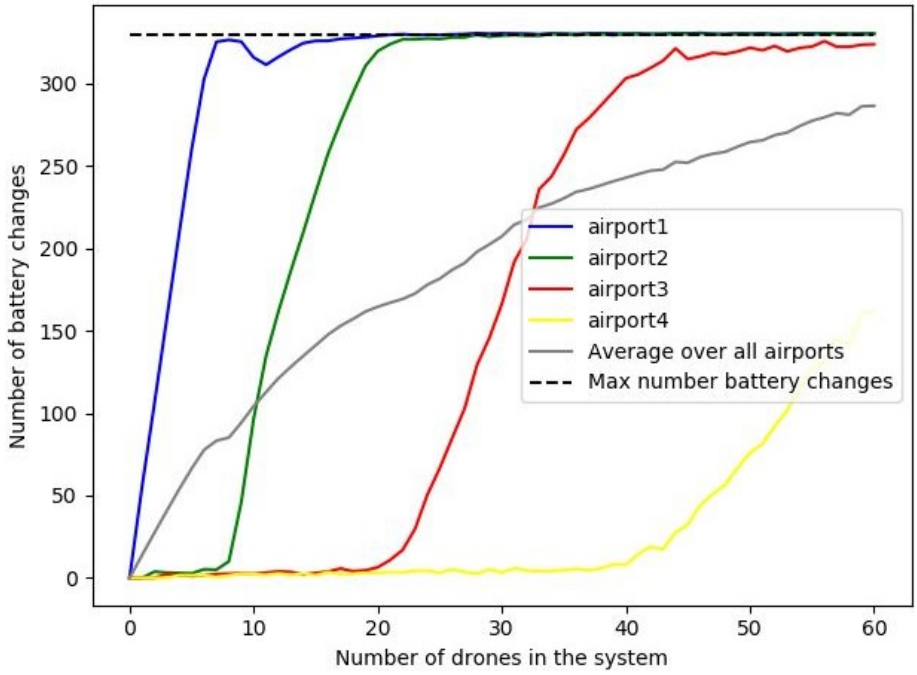


Figure 4.19: Number of battery changes per airport. Swarm optimization, airports with different distances and limited communication.

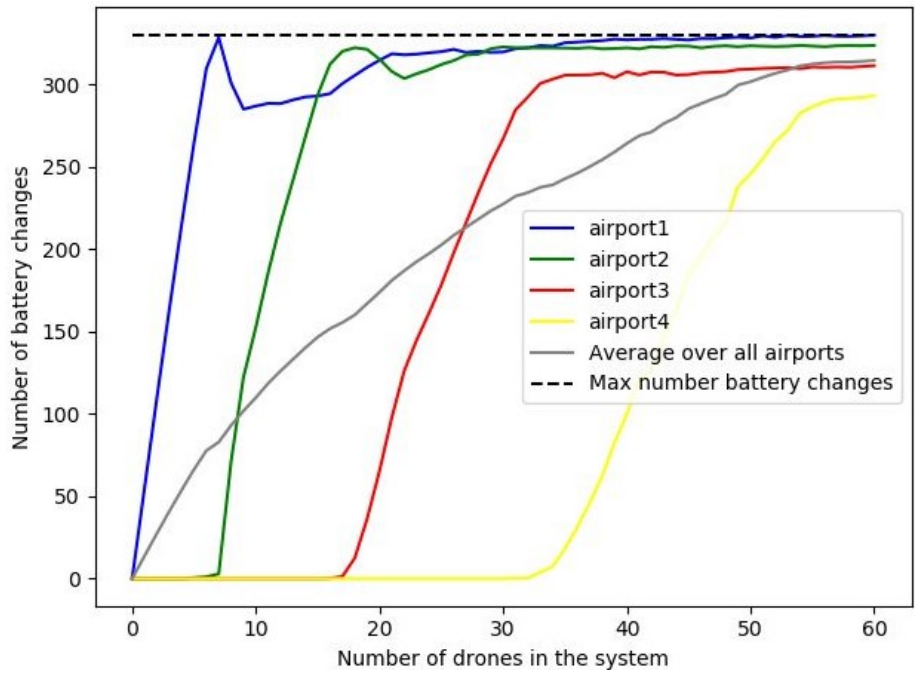


Figure 4.20: Number of battery changes per airport. Swarm optimization, airports with different distances and full communication.

the change of direction occurs at the same time as airport 1 is filled up. It seems like the drones priorities the first airport more than they should, and therefore still choose the first airport even though it would be better to go to the next. One possible explanation can be the same as in the last section, that the information gets "old" by the time a drone uses it to find the next airport. In 4.20 there is also a dip in the graph right after it reaches the maximum number of drones for an airport. This could be an effect caused by the size of the swarm increasing, and information is therefore spread faster. That way, the drones understand that they should go to airport 2 instead. Another explanation can be, as in the last section where several drones end up choosing the same airport at the same time, creating a large queue. When there is a large queue, none of the other drones go to this airport, and, as a result, the airport can suddenly be without drones. The drones still think there is a line even though the airport is finished with the whole queue. A combination of all these effects can be the reason for the sudden change in figure 4.18 explained above.

From about 6-7 drones and to about 20 drones there is a quite linear increasement in figure 4.18 for all the swarm methods. When the system has around 20 drones, the methods with limited communication again have a change in direction, while the methods with full communication continue the apparently linear line. Between 30 and 35 drones, both communication configurations experience a new change of direction in their graph. When comparing these points with figure 4.20 they match up fairly well with when airport 3 and 4 starts being used. For figure 4.19 on the other hand, it does not match as well. It matches with when airport 3 starts being used, but not airport 4. For some reason airport 4 is not used before there are around 40 drones in the system. This again can be explained by the information being "old".

When the system is very over-saturated ($> 40$ drones) the full communication methods (figure 4.20) are able to use all airports increasingly equally, but struggles to use all the potential, resulting in only airport 1 fully converging towards the maximum number of battery changes. The limited communication methods, on the other hand (figure 4.19), are not able to use airport 4 as much, but airport 1, 2 and almost 3 are fully converged towards the maximum.

### 4.5.3 Comparing methods

After looking at all the methods individually in this configuration, they all seem to have there strengths and advantages, but also faults. In figures 4.21 and 4.22, all methods can be seen together. The blue representing the centralized method, green the swarm methods, yellow the random method without distance taken into account and red is the random methods where distance is taken into account. Again, the centralized method had the best

results showing that communication often is an advantage. The results are better in any of the saturation states.

Looking at the methods where there is no centralized control, the random methods perform rather impressively. As shown in figure 4.22, the random method which does not take the distance into acountance follows the centralized method quite well. It has the same form but has a little offset towards the worse. When looking at the percentage like in figure 4.21 it is easier to see that this random method is not as good as it looks in the very under-saturated system, but catches up with the centralized method to a certain extent when there are more drones in the system.

The other random method seems to perform best between 10 and 23 drones. This is the area where it is either better than the other random method (yellow line), better than the swarm methods or better than both. The downside is that it does not handle a over-saturated system well. It never fully engages all airports, and is therefore not able to get as many active drones in the air as seen in figure 4.22. This is also the problem for the swarm methods. Especially the methods with limited communication are not able to achieve a satisfying number of active drones compared to the centralized method and the good random method. The swarm method with full communication, on the other hand, almost achieves the same number of active drones as the random method in yellow.

The swarm methods are only good when the system is very under-saturated. Already around 13-14 drones in the system, the random method in yellow becomes better.

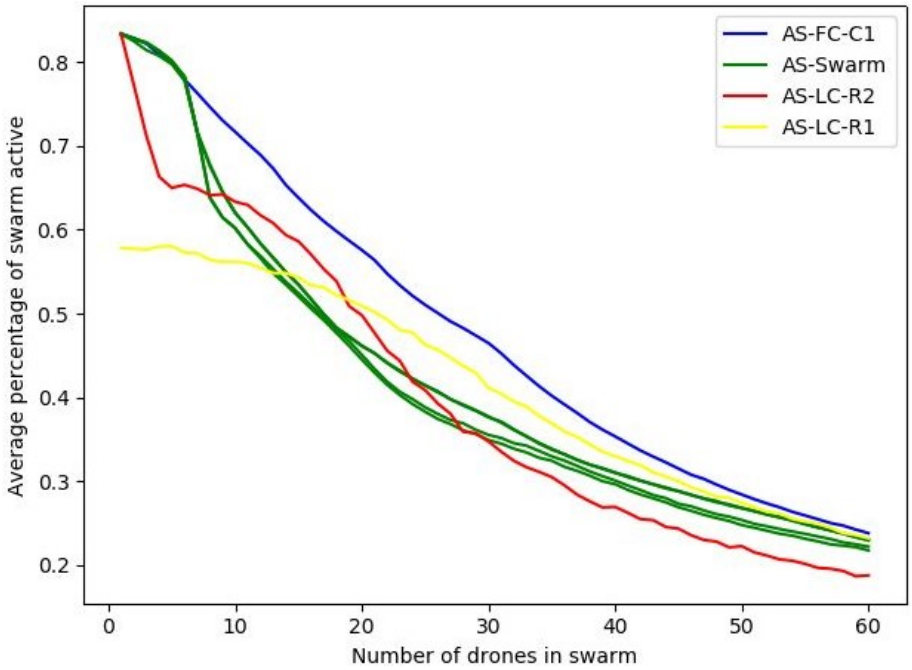


Figure 4.21: Average percentage of drones active: All methods.

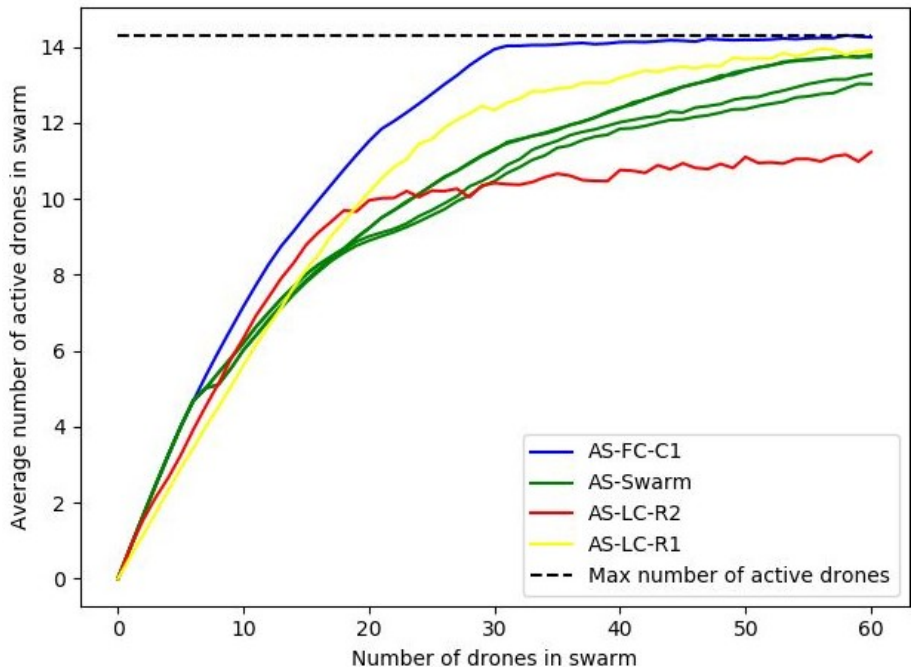


Figure 4.22: Average number of active drones: All methods.

# Chapter 5

# Discussion

Before addressing the research questions from section 1.2, it is important to mention a couple of weaknesses in the thesis. Whether they affect the results in a positive or negative way is hard to say, but they should have been evaluated in the experiments. For instance, the limited communication should have been tested with several degrees of limited, in order to fully understand how the limitation affects the system. This means that the limited communication used in the experiments have not been tested to see if it represents this configuration in an adequate way. Other parameter choices that should have been better tested is the size of the drone workspace, the drone speed, and general battery parameters. All of them affect how far a drone can travel, and in real life, all drones have different configurations of this. Therefore, they should have been better tested before implemented.

It is also important to emphasize that even though the results seem both satisfying and not satisfying they do not represent neither the best or worst possible response threshold method. There is most likely an implementation which can improve the results in both cases and hopefully better defend the usage of swarm intelligence as a solution to the drone swarm airport problem. Possible solutions to this will be discussed in the future work section below.

## 5.1 General discussion

Based on the results presented in chapter 4, this section will try to answer the research questions asked in the introduction.

1. Is it possible to develop a suitable simulation tool to the drone swarm system where it is possible to implement different optimization methods which vary in degree of centralization and communication?

2. Is is possible to optimize a decentralized set of drone swarm airports with a method inspired by response threshold?

As mentioned in chapter 1.2, a simulation tool was a prerequisite in order to be able to implement and test the optimization methods. The short answer to the first research question is therefore yes. The tool made it possible to test different methods and made it possible to vary the amount of communication and centralization. This makes the simulation tool suitable in the way that it fulfills the requirements mentioned. Whether the tool is credible or not is also a question which effects how suitable it is. Based on the results achived compared with the few analyticly calculated results it seems as the simulation tool producess adaquate and credible results.

After analyzing the results from chapter 4, the answer to the second question can be both yes and no. In the experiments where the airports were located at the same place, the results appear to be satisfying compared to the benchmark systems in the system. The method is clearly better than the random method, which means that it improves the performance rather than make it worse compared to the method representing the lower rank. The results also confirm the initial assumption that the centralized method will be better. Moving on to the other configuration where the airports are spread, the results are almost as good as the random method, but based on the results presented it is difficult to accentuate the swarm method as a successful optimization method to the problem.

Since the answer to the research question is based on comparing the results with the benchmark methods on must also ask if they are adquate in terms of validating the results of the swarm method. There is no way to asserain whether the benchmark systems are adequate or not, but it is possible to present arguments that strengthen the assumptions that they are. There are two benchmark systems that represent the outer edges of the solutions. The centralized method, which tries to show how good the system can work, seem to perform rather well based on the result analyzed in the previous chapter. Since the goal of the method is for it to be better than the swarm method, it has indeed achieved this, but whether or not it is representing the better solution in a good matter can be debated. It could be the case that the best solution is a lot better resulting in the swarm method looking even worse, but this would not affect the main research question where the more important comparison is against the random method. Therefore the centralized method is evaluated as an adequate benchmark method.

When it comes to the random method, it also seems to be adequate. The goal of the method is to represent a solution that is a realistic lower limit of what is acceptable performance. If the random method had performed really bad the sufficiency could have been questioned, but based on the results and assuming that the implementation is realistic, the method seems adequate. In fact, the method performed better than expected. The results prove that a random approach can actually be a good solution.

## 5.2 Conclusion

This thesis has explored possible ways of optimizing a drone airport system. Since the system, in general, seem to be a rather new contribution to any research field, this thesis has also introduced one way of looking at an autonomous drone airport system. As a result, the thesis has presented 3 different optimization methods where the goal for the methods is to choose which airport a drone should go to when it needs to change its battery.

**A simulation tool** was created in order to test the different methods and being able to compare the methods, knowing that the configuration and setup are equal. The simulation tool was created in Python with standard scientific packages. To be sure that the simulation worked as it should, a visualization tool was created using a package called "Pygame". The visualization tool was an extra feature only used when the goal was to understand what happened in the simulation and was not used when running big experiments, due to the time consumption of rendering the environment.

**The swarm method** introduced is inspired by the existing response threshold methods within the science of swarm intelligence. The goal of this method was to create a solution that is independent of communication and not in the use of centralized control. The method is implemented in the simulation tool which resulted in positive and negative results but without a doubt showing that the method is interesting to study further.

**The centralized methods** were created as benchmark systems, but are also new to this specific problem. The first method introduced as centralized is a mathematical optimization solution inspired by linear programming. The method implemented was used to show how well one airport could be optimized, but also confirmed the assumption that it is computational heavy when increasing the size of the system. Therefore, there was also created another centralized method that was implemented when simulating several airports. This method had the same properties regarding centralization and communication but is not proven to be mathematically optimized. This method is also a new contribution and shows that it is possible to implement a rather good solution to the problem of interest.

Finally, **the random methods** represent the opposite of the centralized method. The goal of the method was to understand how well the system works with no logical or smart decision making. The random method was also implemented in the simulation tool and compared to the other two methods. The method resulted in performing rather well and showed that it is a proper challenger to the other methods.

## 5.3 Future work

This thesis contributes to a research field that without a doubt is yet to be fully explored. The amount of future work can seem endless, but below are some of the more specific possibilities related to the thesis.

**Improve swarm method -** The swarm method implemented had ambiguous results which most certainly have the potential for improvement. Below is a list of some possible solutions to achieve such an improvement:

1. Tune parameters by running more testes, varying the parameters and comparing the results. This could improve a little bit, but is not believed to make any big difference.

2. Update the information flow with a smarter solution. One possible solution is to let the drones spread more information, not only about themselves but also the information they have received from other drones. If the drones always spread their updated overview list, the receiver could update information for all airports and not only the single airport information.

3. Including the airport in the communication flow. If the airport is implemented with the same opportunity for communication as the drones, the drones closest can always get the newest information. If this is implemented with the point mentioned above, the newest information can, in theory, reach all drones even though they are far away.

4. Change the stimuli for the drones. It is a possibility that one can find information which is more informative and therefore is a better choice. One example could be to include how busy an airport has been, and not only how busy it will be.

5. Update the calculations done when calculating the response. This point is connected with point 4 and can be forced to change if the stimuli changes, but it can also be a possible solution to study a better way of calculating with the same information as in this thesis.

**Airports with different efficiency -** By implementing either airports which can take more drones at the same time, or implement diversity in the efficiency will introduce another dimension when deciding an airport. It is also more likely in a real-world system that the airports have a more variable time consumption on a battery change.

**More advanced airports -** In the existing implementation the airports are very passive, and do nothing else but change the battery when a drone

arrives. Below is a list of possible ways to increase the service and general flow at the airport:

1. Expanding the service done at the airports. There are more parts of the drone which needs to be changed, just not as often. This could be propellors, motors, cameras, other equipment, packages, etc. This would affect the time consumption for a service considerably, making it much more difficult to calculate when it is available. This change would also make the system even more autonomous. The airports could also have different tasks, meaning that the drones will go to another airport for a battery change compared with a motor change and a third for propellor change. The airports could also offer several services at once, where each airport has its unique combination of possible services.

2. Prioritizing the drones. Based on what the system is used for the drones could have different missions, where some missions are more important than others. If this is the case, the airports could prioritize which drones that get service first based on the priority of their mission.

3. Prioritizing based on payment. This idea is inspired by the priority idea in the previous point. It could be possible that the airports are not part of a specific system, but operate as an airport we know today. This means that drones and airports are not part of the same system and drones can use airports that are placed at known locations. When a drone arrives it can pay for a service to be done and the airport performance. An example of such a system is grocery delivery. If all the different companies delivering groceries start delivering using drones, a third party can place service airports at several locations that all companies can use. The airport could then prioritize the incoming drones based on who pays the most.

# References


[1] Ilan Adler. "The equivalence of linear programs and zero-sum games". In: *International Journal of Game Theory* 42.1 (Feb. 1, 2013), pp. 165–177. ISSN: 1432-1270. DOI: 10.1007/s00182-012-0328-8.

[2] A. Agustı´n et al. "On air traffic flow management with rerouting. Part I: Deterministic case". In: *European Journal of Operational Research* 219.1 (May 16, 2012), pp. 156–166. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2011.12.021.

[3] C. Anderson, J.J. Boomsma, and III Bartholdi J.J. "Task partitioning in insect societies: bucket brigades". In: *Insectes Sociaux* 49.2 (May 1, 2002), pp. 171–180. ISSN: 1420-9098. DOI: 10.1007/s00040-002-8298-7.

[4] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, June 12, 2013. 1122 pp. ISBN: 978-1-118-62630-6.

[5] Julia A. Bennell, Mohammad Mesgarpour, and Chris N. Potts. "Airport runway scheduling". In: *4OR* 9.2 (June 1, 2011), p. 115. ISSN: 1619-4500, 1614-2411. DOI: 10.1007/s10288-011-0172-x.

[6] Christian Blum and Xiaodong Li. "Swarm Intelligence in Optimization". In: *Swarm Intelligence*. Ed. by Christian Blum and Daniel Merkle. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 43–85. ISBN: 978-3-540-74088-9 978-3-540-74089-6. DOI: 10.1007/978-3-540-74089-6_2.

[7] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. "Quantitative Study of the Fixed Threshold Model for the Regulation of Division of Labour in Insect Societies". In: *Proceedings: Biological Sciences* 263.1376 (1996), pp. 1565–1569. ISSN: 0962-8452.

[8] Interval Research Fellow Eric Bonabeau et al. *Swarm Intelligence: From Natural to Artificial Systems*. Google-Books-ID: PvTDhzqMr7cC. OUP USA, 1999. 324 pp. ISBN: 978-0-19-513159-8.

[9] Glenn A. Bowen. "Naturalistic inquiry and the saturation concept: a research note". In: *Qualitative Research* 8.1 (Feb. 1, 2008), pp. 137–152. ISSN: 1468-7941. DOI: 10.1177/1468794107085301.

[10] Eduardo Castello et al. “Foraging optimization in swarm robotic systems based on an adaptive response threshold model”. In: *Advanced Robotics* 28.20 (Oct. 18, 2014), pp. 1343–1356. ISSN: 0169-1864. DOI: 10.1080/01691864.2014.939104.

[11] Grégoire Chamayou. *Drone Theory*. Google-Books-ID: PL8cBAAAQBAJ. Penguin UK, Jan. 29, 2015. 241 pp. ISBN: 978-0-241-97035-5.

[12] GB Dantzig. “A proof of the equivalence of the programming problem and the game problem”. In: *Koopmans TC (ed) Activity analysis of production and allocation* (), pp. 330–335.

[13] Daniel Delahaye and Stéphane Puechmorel. “Air Traffic Control”. In: *Modeling and Optimization of Air Traffic*. John Wiley & Sons, Ltd, 2013, pp. 83–90. ISBN: 978-1-118-74380-5. DOI: 10.1002/9781118743805.ch4.

[14] Marco Dorigo and Erol Şahin. “Guest Editorial”. In: *Autonomous Robots* 17.2 (Sept. 1, 2004), pp. 111–113. ISSN: 1573-7527. DOI: 10.1023/B:AURO.0000034008.48988.2b.

[15] Jurgen Eichberger. “Review of Game Theory, , ; Game Theory: Analysis of Conflict”. In: *The Economic Journal* 103.419 (1993). In collab. with Drew Fudenberg, Jean Tirole, and Roger B. Myerson, pp. 1065–1067. ISSN: 0013-0133. DOI: 10.2307/2234726.

[16] Huilian Fan. “A Rough Set Approach to Feature Selection Based on Wasp Swarm Optimization”. In: 2012.

[17] Saul I. Gass. “George B. Dantzig”. In: *Profiles in Operations Research: Pioneers and Innovators*. Ed. by Arjang A. Assad and Saul I. Gass. International Series in Operations Research & Management Science. Boston, MA: Springer US, 2011, pp. 217–240. ISBN: 978-1-4419-6281-2. DOI: 10.1007/978-1-4419-6281-2_13.

[18] S. Gupte and {and} J. M. Conrad. “A survey of quadrotor Unmanned Aerial Vehicles”. In: *2012 Proceedings of IEEE Southeastcon*. 2012 Proceedings of IEEE Southeastcon. Mar. 2012, pp. 1–6. DOI: 10.1109/SECon.2012.6196930.

[19] Heiko Hamann. *Swarm Robotics: A Formal Approach*. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-74526-8 978-3-319-74528-2. DOI: 10.1007/978-3-319-74528-2.

[20] James Kennedy. “Particle Swarm Optimization”. In: *Encyclopedia of Machine Learning*. Springer, Boston, MA, 2011, pp. 760–766. DOI: 10.1007/978-0-387-30164-8_630.

[21] Min-Hyuk Kim, Hyeoncheol Baik, and Seokcheon Lee. “Response Threshold Model Based UAV Search Planning and Task Allocation”. In: *Journal of Intelligent & Robotic Systems* 75.3 (Sept. 1, 2014), pp. 625–640. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-013-9887-6.

[22] David G. Luenberger and Yinyu Ye. "Basic Properties of Linear Programs". In: *Linear and Nonlinear Programming*. Ed. by David G. Luenberger and Yinyu Ye. International Series in Operations Research & Management Science. New York, NY: Springer US, 2008, pp. 11–31. ISBN: 978-0-387-74503-9. DOI: 10.1007/978-0-387-74503-9_2.

[23] Brian M. Mann and John W. Poore. "Microprocessor controlled rate-responsive pacemaker having automatic rate response threshold adjustment". U.S. pat. 4940052A. Pacesetter Inc. July 10, 1990.

[24] Elise L. Mansfield et al. "Adjustments of Response Threshold during Task Switching: A Model-Based Functional Magnetic Resonance Imaging Study". In: *Journal of Neuroscience* 31.41 (Oct. 12, 2011), pp. 14688–14692. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.2390-11.2011.

[25] Janice M Morse. *The significance of saturation*. May 1995.

[26] Roger B. Myerson. *GAME THEORY*. Harvard University Press, Mar. 1, 2013. 585 pp. ISBN: 978-0-674-72861-5.

[27] Kenzo Nonami et al. *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*. Springer Science & Business Media, Sept. 15, 2010. 341 pp. ISBN: 978-4-431-53856-1.

[28] *Nonlinear programming*. In: *Wikipedia*. Page Version ID: 865897558. Oct. 26, 2018.

[29] Olav Rune Nummedal et al. *Development and construction of a low-cost autonomous battery replacement robot for drones*. 2017.

[30] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. Google-Books-ID: mnv1DAAAQBAJ. MIT Press, July 12, 1994. 369 pp. ISBN: 978-0-262-65040-3.

[31] University of California Agriculture {and} Natural Resources. *California Forests*. URL: http://ucanr.edu/sites/forestry/California_forests (visited on 05/29/2018).

[32] Gene E. Robinson. "Modulation of alarm pheromone perception in the honey bee: evidence for division of labor based on hormonall regulated response thresholds". In: *Journal of Comparative Physiology A* 160.5 (Sept. 1, 1987), pp. 613–619. ISSN: 1432-1351. DOI: 10.1007/BF00611934.

[33] Gene E. Robinson. "Regulation of Division of Labor in Insect Societies". In: *Annual Review of Entomology* 37.1 (Jan. 1, 1992), pp. 637–665. ISSN: 0066-4170. DOI: 10.1146/annurev.en.37.010192.003225.

[34] Jan A. Snyman and Daniel N. Wilke. "INTRODUCTION". In: *Practical Mathematical Optimization: Basic Optimization Theory and Gradient-Based Algorithms*. Ed. by Jan A Snyman and Daniel N Wilke. Springer Optimization and Its Applications. Cham: Springer International Publishing, 2018, pp. 3–40. ISBN: 978-3-319-77586-9. DOI: 10.1007/978-3-319-77586-9_1.

[35] Daniel Solow. "Linear and Nonlinear Programming". In: *Wiley Encyclopedia of Computer Science and Engineering*. American Cancer Society, 2007. ISBN: 978-0-470-05011-8. DOI: 10.1002/9780470050118.ecse219.

[36] Ricardo Soto, Eduardo Rodriguez-Tello, and Eric Monfroy. *Recent Advances on Swarm Intelligence for Solving Complex Engineering Problems*. Mathematical Problems in Engineering. 2018. URL: https://www.hindawi.com/journals/mpe/2018/5642786/ (visited on 04/28/2019).

[37] Maarten Tielrooij et al. "Supporting Arrival Management Decisions by Visualising Uncertainty". In: (2013), p. 9.

[38] *Topic 5: Linear Programming*. URL: http://jwilson.coe.uga.edu/emt668/EMAT6680.2002/Jackson/EMAT%206000/topic%205/topic%205-lin%20program.html (visited on 04/07/2019).

[39] Robert J. Vanderbei. *Linear programming: foundations and extensions*. New York: Springer, 2013. ISBN: 978-1-4614-7629-0.

[40] Edward O. Wilson. "Caste and division of labor in leaf-cutter ants (Hymenoptera: Formicidae: Atta)". In: *Behavioral Ecology and Sociobiology* 7.2 (July 1, 1980), pp. 143–156. ISSN: 1432-0762. DOI: 10.1007/BF00299520.

[41] Edward O. Wilson. "Division of Labor in Fire Ants Based on Physical Castes (Hymenoptera: Formicidae: Solenopsis)". In: *Journal of the Kansas Entomological Society* 51.4 (1978), pp. 615–636. ISSN: 0022-8567.

[42] Edward O. Wilson. "The relation between caste ratios and division of labor in the ant genus Pheidole (Hymenoptera: Formicidae)". In: *Behavioral Ecology and Sociobiology* 16.1 (Nov. 1, 1984), pp. 89–98. ISSN: 1432-0762. DOI: 10.1007/BF00293108.

[43] Michael Wooldridge. *An Introduction to MultiAgent Systems*. Google-Books-ID: X3ZQ7yeDn2IC. John Wiley & Sons, June 22, 2009. 484 pp. ISBN: 978-0-470-51946-2.