UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

VINÍCIUS RORATTO CARVALHO

# A comparative study on multi-UAV approaches for crowd monitoring systems

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering

Advisor: Prof. Dr. Edison Paschoal Gaspary

Porto Alegre
Setembro 2022

## ACKNOWLEDGMENTS

To my wife and my parents who helped me through every step of this course.

## ABSTRACT

This study compares artificial intelligence algorithms in solving the problem of crowd monitoring by multiple unmanned aerial vehicles (UAVs). The techniques used to solve the problem are the genetic algorithm (GA), the ant colony optimization (ACO) , simulated annealing (SA) and immune algorithm (IA) . The framework used for simulating the agents is the Robot Operating System. The study found out that in this implementation of the problem, no specific routing algorithm showed significant advantages over the others.
**Keywords:** Crowd Monitoring. Unmanned Aerial Vehicles. Genetic Algorithm. Ant Colony Optimization. Robotic Operating System.

**Um estudo comparado em diferentes abordagens de monitoramento de multidões utilizando múltiplos VANTs**

## RESUMO

Este estudo compara algoritmos de inteligência artificial na resolução do problema de monitoramento de multidões por múltiplos veículos aéreos não tripulados (VANTS). As técnicas utilizadas para resolução do problema são o algoritmo de evolução genética (GA) e a otimização de colônias de formigas (ACO), recozimento simulado (SA) e algoritmo imune (IA) no framework Robot Operating System (ROS). O estudo conclui que, na implementação corrente, nenhum algoritmo apresenta vantagem significativa sobre os outros.

**Palavras-chave:** Monitoramento de Multidões, Veículo aéreo não tripulado, Algoritmo genético, Otimização de colônia de formigas, recozimento simulado, Algoritmo Imune.

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| ACO | Ant Colony Optimization |
| AI | Artificial Intelligence |
| GA | Genetic Algorithm |
| IA | Immune Algorithm |
| MAS | Multi-Agent Systems |
| SA | Simmulated Anealing |
| UAV | Unmanned Aerial Vehicle |
| VANT | Veículo Aéreo Não Tripulado |

# LIST OF SYMBOLS

| | |
|---|---|
| $\sum$ | Sum |
| $\sigma$ | Standard Deviation |
| $f$ | Function |

# LIST OF FIGURES

# LIST OF TABLES

**CONTENTS**

# 1 INTRODUCTION

The main goal of this project is to test how different routing algorithms perform to solve an unmanned aerial aircraft (UAV) team management application problem. The project simulated 12 different scenarios of crowd control where the artificial intelligence (IA) manages resources, assigning responsibilities to different agents while conditions in the environment change.

The research of this type of application has become important to the community because the recent reduction in cost has popularized the use of UAVs in several activities (MORAES; FREITAS, 2020). They include military applications, law-enforcement, agriculture, environmental monitoring, engineering solutions and can be extrapolated to several more.

Despite its versatility, several of the most common applications still need to be manually controlled or have its flight plan previously defined. This really limits the possibilities of the tool in a chaotic environment, like a crowd, when the amount of outcomes cannot be predicted by a simple flight plan.

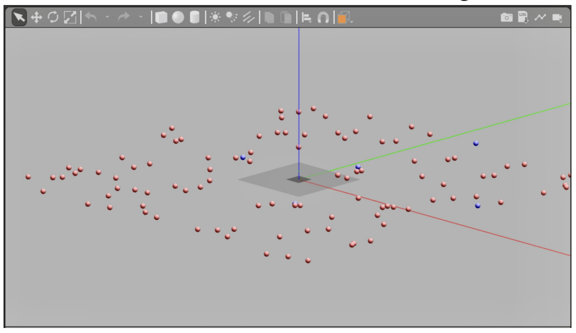
There is another important limitation: in crowd monitoring scenarios, security forces usually have limited staff, but can be faced with many targets to follow . One machine, controlled by one remote pilot, will do very little to improve the monitoring of several moving targets in a crowd (MORAES; FREITAS, 2020). To find one solution to this challenge, this work suggests the use of multiple UAVs managed not by a team of pilots, but by artificial intelligence techniques.

This project investigated the problem of tracking the position of many targets with multiple agents. This is not a new problem, it's been described in past bibliography, like Moraes and Freitas (2020), where it uses UAVs to handle a varying number of targets in a crowd, implementing the solution, the environment and several different tests in an ad hoc platform.

The main goal here was to fully rebuild the solution inspired by Moraes and Freitas (2020) for an open source robotics simulation environment, compare how different routing algorithms perform within the solution and make it available online under a creative commons licence, where other engineering students can improve on the work that have been done.

The solution proposed is tracking and recording the position or the approximate location of a number of targets bigger than the number of agents in a crowd. The UAVs should continually hover between targets to keep track of their position in a wide, open, unobstructed area.

Figure 1.1 – Environment simulation with red dots as targets and blue dots as UAVs.



Source: Ros-Gazebo Simulation

There are some attributes that need to be respected to perform a similar simulation: the crowd must be pacific, having no sudden changes of speed, with a few potential targets. The UAVs know the number and location of the targets in the beginning of the simulation. There are several problems to be solved in these conditions, this project will test only how the agents behave to monitor the targets, image processing to identify the targets and movement control are not on the scope.

The final goal of the simulation is to, within limitations, reproduce real situations and possible environments, like demonstrations in front of government buildings. Events where the crowd has limited movement freedom even if in a large, unobstructed terrain and certain specific targets can be monitored by air are easily translatable to protests in the United States' Capitol (Washington, D.C.), in the Brazilian Esplanada dos Ministérios (Brasília) or in the Egyptian Tahrir Square (Cairo).

# 2 CONCEPTS REVIEW

Before analysing the full problem, it's important to understand the basic concepts that grounded this paper. The next subsections describe what contemporary research understands as robots, unmanned aerial vehicles, artificial intelligence and multiagent systems.

## 2.1 Robots

In its most basic form, an unmanned aerial aircraft can be defined as a mobile robot, Garcia et al. (2007) describes it as a "system able to carry out tasks in different places", its mobile elements will depend on its operation environment. As mobility improves, the range of capabilities of the system becomes more complex.

In 1968, the first industrial mobile robots were introduced in factories, they were basically automated vehicles following predefined paths along the plant (GARCIA et al., 2007). The more modern research in the area deals with indoors and outdoors autonomous navigation. Garcia et al. (2007) also points out four stages necessary for robot navigation: "perception of the environment, self-localization, motion planning and motion generation".

One of the biggest challenges in the area is robot navigation. It has two very different scenarios: structured environments that can be modeled beforehand and unstructured dynamic environments where the robot needs to use its sensors to understand its position and learn to move on the map (GARCIA et al., 2007).

## 2.2 Unmanned Aerial Vehicles

There are two very distinguishable formats of unmanned aerial vehicles, first the fixed wing model, built with simple structures based on efficient aerodynamics, but limited by the necessity of runaways and need to move constantly not to fall. The second model is the rotary wing, or multi-rotor, UAV, popularly known as heli, tetra, hexa or octa copters depending on the number of rotors in its project. When this project mentions UAVs, it's referring to a multi-rotor model (TIWARI, 2020).

Both models can be controlled remotely by a ground pilot, via remote-control (teleoperation), or autonomously. The multi-rotor models have some advantages that can be explored, and some issues that become part of the challenges to be overcome: the ability to take-off and land vertically and operating in a limited space, performing quick speed and direction changes can be very useful to the simulations proposed, but the shorter flight duration can really restrain the types of missions where it can be used (TIWARI, 2020).

These UAV attributes, if coordinated correctly, can be very useful for the scenarios suggested in the simulation. Moraes and Freitas (2020) describe a system developed to manage the monitoring responsibilities of not just one, but multiple vehicles, focusing on minimizing the time each target remains with no surveillance.

Tiwari (2020) identifies some advantages and challenges of working with multiple robots:

- Distributed computing: how much processing one drone alone can handle will always be limited to their hardware, but when a team is available, the load can be distributed with different strategies, easing the computational burden of a single unit.
- Wider area coverage: there are different constraints that can prevent a single unit to cover wider areas, battery and mission time are two examples. A team can improve the performance just by being able to navigate to more areas at the same time.
- Temporally optimal coverage: in time-constrained missions, it is possible to improve results by using a team of robots that can maximize data recollection performance locally.
- Heterogeneous coverage: This advantage is about specialization. It's possible to improve final mission results by combining strengths of different models. A simple example would be using a quick small model to find a target, then communicate the way to a bigger, more specialized robot that could carry provisions to the targeted location.
- Using more than one UAV means some decision on how to manage the team will have to be taken before a mission starts. For example, mission strategies can be centralized, one node will make decisions, decentralized, with multiple decision-making points or distributed, where there is a collective equal work to decide on actions.

These same strategies can change how the group can communicate: data flows between the nodes can be synchronized, unsynchronized or even totally disconnected. There is not just one right answer for these decisions, they must be done taking into account the resources available and the goals of the mission. And the bigger the team gets, the more complex the decisions will be.

It is possible to look at the features of UAVs described above and start describing the solution proposed in this project. Since all UAVs are the supposed to be the same, and will handle different targets delivered by an outside source, this is a homogeneous-centralized mission that will distribute its responsibilities to improve temporal and area coverage. Also, each UAV will process its own routes internally, working as a distributed computing system.

## 2.3 Multi-Agent Systems

An interesting way to examine these Multi-agents systems (MAS) is to compare them to computing entities in a distributed problem-solving structure, where the interactions with neighbours can help them improve their autonomous decisions (DORRI; KANHERE; JURDAK, 2018).

Some advantages of using MAS for problem-solving can be very easy to point out, like the possibility of labour division and specialized task assigning between the agents. But there are others like higher reliability, the possibility of easily substituting defective agents, simpler scalability (DORRI; KANHERE; JURDAK, 2018).

There are a number of attributes that must be considered when designing a MAS, for example, the level of sociability between the agents, or how they share knowledge and request data from other agents (DORRI; KANHERE; JURDAK, 2018), can they have the ability to decide when to act? Can they act differently depending on the levels of knowledge and perception of other agents?

Dorry, Kanhere and Jurdak (2018) help to describe a number of features that can be listed to help evaluate decisions on designing a MAS:

- Leadership: a system can be leaderless or leader-follow. There can exist more than one leader, and they can be mobile and communicate in different ways to the followers.
- Decision Function: can be either linear or nonlinear and depends directly on the decision function, for example changing decisions based on wind speed can be

following a linear threshold, while making a decision based on face recognition may be a nonlinear function.

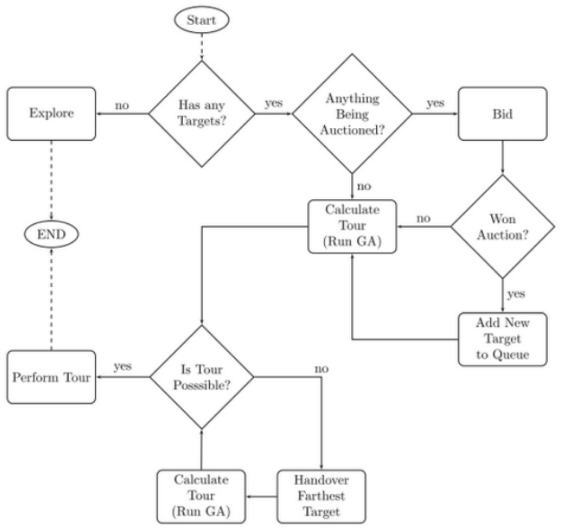
- Heterogeneity: homogeneous groups have all the same characteristics, while heterogeneous groups can include a diversity of features.
- Agreement Parameters: First order agents need to agree on only one metric, second order agents need to agree on two metrics to succeed on its goals.
- Delay Consideration: the delay on information distribution can be taken into account depending on the metrics relative to the goals. A group of autonomous vehicles that scout one area until their batteries are low, and then coming back to the base have no consideration on what the delay should be. If they have a trigger and have to all come back together once an event happens, the delay between the communications can be important to the mission.
- Topology: In a static topology MAS the relationship between the agents never change, in a dynamic topology some agents can leave or join the MAS.
- Data Transmission Frequency: how quick agents must share their knowledge of the environment can be defined either by time or by event.

In this project, the UAVs don't communicate directly between them, but to a system that assigns them the targets, so we can describe the leader as an outside UAVs resource management system. The internal decision functions in each UAV take use two parameters to make decisions: target positions and speed. The agents are homogeneous, and the topology is static, because their relations are fixed from the beginning to the end of the test. Communication speed and frequency is not something measured in this project because it's not under this project's scope.

# 3 RELATED WORK

Since this paper seeks to test already simulated results against other AI algorithms, there is already a body of work that helps understand the challenges. In Moraes and Freitas (2020), for example, the basic problem of periodically monitoring a crowd with multiple UAVs is described, and one solution is suggested: auctions to distribute targets and genetic algorithms to calculate visiting order.

Figure 3.1 – Moraes and Freitas solution involves a bidding, a routing and a target delivery system.



Source: (MORAES; FREITAS, 2020)

Another study by the same authors (MORAES; FREITAS, 2018) also suggest another strategy to coordinate the UAVs, studying how hybrid algorithms merge ant colony optimization, auctions and proactive link maintenance, to perform in surveillance missions. The problem suggested by the article has one extra complication: the agents must maintain communications between the group during the mission, and once they find their points of interest, they should deliver a mission to another assigned agent. This paper focuses specifically on coordinating the teams, communications and different agent handling are not on the scope.

Freitas et al. (2010) addresses strategies of coordinating a team of UAVs with a sensor network, like target sensing alarms. The proposed approach compares a decentralized decision-making method to a centralized strategy, showing that the first lower communication costs. In this coordination strategy, any UAV that receives an alarm from the sensors have to negotiate with the other agents in the mission. Also, different agents can have different profiles, changing their ‘chances’ to be interested to engage in the activity depending on associated functions (FREITAS et al., 2010).

Khan, Rinner and Cavallaro (2018) review research of simultaneous robot coordination when monitoring moving targets. The paper presents the elements that define the coordination problem: the coordination method, the environment, the target, the robot and its sensors. The control techniques presented depend on target behaviour: cooperative tracking, the idea of using frequent updates on the target situation by the creation of fixed cooperative tracks; Cooperative multi robot observation of multiple moving targets, where there is the necessity of searching for targets that can walk away from predefined paths. Cooperative search, acquisition and track: where the team needs to constantly search for unknown targets. And, to finish, multi robot pursuit evasion: where the targets are aware of the agents and can evade them (KHAN; RINNER; CAVALLARO, 2018).

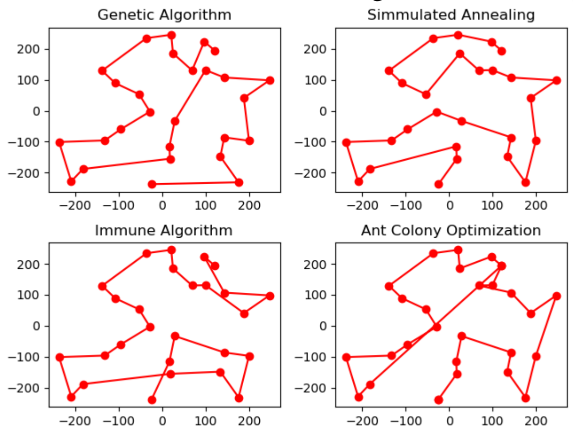
Ergezer and Leblebicioglu (2013) approaches the problem of designing the paths vehicles are supposed to follow to maximize data collection, avoid forbidden regions and reach final destinations. The research suggests a path planning solution using Genetic Algorithms (GA). The goal is to build a series of operators with the capability of executing actions like pulling to a region, pulling out of a region and pulling to a final point (ERGEZER; LEBLEBICIOGLU, 2013).

Brambilla et al. (2013) focuses on swarm robotics and taking inspiration from self-organizing social animals and discusses the limits of the discipline, proposing a taxonomy to classify works according to the behaviours studied. Dorry, Kanhere and Jurdak (2018) on the other hand, study multiagent systems as a means of dividing problems into simpler tasks. The paper describes different aspects of working with MAS, from features and applications to its challenges.

# 4 PROPOSED SOLUTION

To keep the results generally comparable to Moraes and Freitas (2020) it's necessary to follow the same path to their results. That means maintaining the parts of the solution that will not be compared, like the auction algorithm to distribute targets and the handover techniques for agents that need to let go of a target, and then compare the four different algorithms when it comes to building the routes through the targets. It's important to remind that this project is a full reimplementation in a different environment with different programming languages, so no absolute numbers will be compared, just overall tendencies.

Figure 4.1 – First test with different algorithms used in project



Source: Project data.

The original project implemented a mechanism where bidders (UAVs) submit sealed proposals and the decision of the winner is based on a value (bid) formed by a set of parameters based on targets and agents state. In this case, the parameters are the distance to the target, the number of targets already being monitored by the UAV and the speed of the target. The full auction equation is described below at implementation.

The handover algorithm is activated once the UAV realizes it won't be possible to visit all the targets assigned to it without losing the location of one or more. Once a target is flagged as not possible to achieve, the agent calls for a bidding process in which it can not participate, and hands over the target to the winner. The targets offered in bids are the farthest from the other targets.

The routing algorithms used by the UAV to navigate targets are described below:

## 4.1 Genetic Algorithm

The Genetic algorithm (GA) suggested by Moraes and Freitas (2020) will control the order each agent will visit the targets. The main goal is to build a path where no target can move enough to evade the monitoring between two visits. GAs use an initial set of possible solutions to a problem, and then evolve them until one of the solutions can be considered fit by the algorithm (HOLLAND, 1975). They are effective to generate paths to UAVs, and are used, for example, to design a path to a vehicle maximizing information gathering and still avoiding forbidden/obstructed regions (MORAES; FREITAS, 2020).

The objective function described as $min(average(f_a, f_b, f_c, f_d, ...))$ (MORAES; FREITAS, 2020) where $(f_a, f_b, f_c, f_d, ...)$ is the maximum distance moved by a target in between two visits. The genetic solver creates 25 possible solutions to each UAV, assigning targets randomly. The result is an array expressing the visitation order to that specific solution. Analysing each solution of the array according to a fitness function described as $Fa = max(d_1, d_2, d_3)$, where the solution in the array and $d_1, d_2, d_3$ is the distance each target travelled between the last visit, the fitness of that particular function will be calculated as $F_{sol} = average(F_a, F_b, F_c, ...)$. With all the 25 solutions analysed, three solutions will be chosen to minimize $F_{sol}$. This is the first iteration.

The next iteration will choose the two best fit of the last one, copy each of them 12 times (to a total of 24) and choose a totally random one as the 25th. The copied functions will be submitted to a random mutation in 20 percent of its solutions, to keep the bigger part of the best solutions, but still offer some differences to be tested in the iteration. Solutions are reanalysed, and the process repeats for a total of 5 runs. By the end of the process, the function with the best fit is the course set to the UAV in this round of visits.If the solution is not possible, the handover algorithm is called to reduce the list of targets.

## 4.2 Ant Colony Optimization

Another routing algorithm used in this paper is the ant colony optimization algorithm (ACO) suggested by Moraes and Freitas (2018). The chosen algorithm must be able to provide maximum performance of the individual agents while dealing with collision avoidance and environment exploration.

The ACO works through a pheromone map, where agents have the ability to deploy and recognize signals (markers) from other agents in the map. These markers help generate group behaviour, informing other agents through the environment with a low communication cost. Every time an agent senses a marker, it processes it, and chooses its next steps based on that information.

One easy example of using markers is movement. UAVs can facilitate movement where they have already passed by leaving attractive or repulsive markers, pointing other agents to points of interest in the map. Moraes and Freitas (2018) describe the steps for implementing the algorithm in a UAV mission. First, the area of interest is divided into squared sectors, with an area big enough to be processed by the UAV while passing by it. The sectors are the pheromone containers.

Each pheromone unit dispensed in the environment creates a potential field that can attract agents to a certain point or push them away, helping the UAVs to decide their next iteration, following the gradients to points of interest. This pattern can also be used to avoid collisions, by creating repulsive effects around UAVs.

The magnitude of the field suggested by Moraes and Freitas (2018) is proportional to a constant representing the strength of the marker, the distance from the marker to the agent and inverse to the time since the marker was deployed. Using that set of variables, it's possible to assume forbidden areas, like walls, can be avoided through the use of a negative constant value for strength. Also, points of interest can attract several agents in its area through a higher value.

After adding all the potential fields acting in an agent in one moment of time, the next movement is defined and directed and the UAV is pointed to one of its contiguous sectors. The actual movement will depend on the agent's speed and manoeuvring capacity.

## 4.3 Simulated Annealing

The optimization technique known as Simulated Annealing is based on a physical process. Jianlan, Yuquiang and Xuanzi (2010) describe the three steps of the operation: first, a solid material needs to be heated until it becomes liquid, raising the energy of the system and increasing entropy. The second state is the isothermal, where the system will act spontaneously, changing heat directly with the environment, reaching a balance once the energy achieves a minimum. And last, cooling, where the "particle thermal motion decreases and gradually orderly, so that causes the energy down gradually to get to the crystalline structure of energy" (JIANLAN; YUQIANG; XUANZI, 2010). The controlled cooled crystalline material can have different features from the initial material.

Inspired by the same process, the Simulated Annealing (SA) algorithm can be described as a “process starts from initial solution i and control parameter initial value t, secondly, repeats iteration for current solution with process that generate new solution, calculate different value of target function, accept or abandon, then gradually attenuation value of t” (JIANLAN; YUQIANG; XUANZI, 2010). The heuristic seeks to find optimum global values for the repeated function. A step by step explanation to the algorithm would be something like this: manually, or randomly, an initial solution to the evaluated function is chosen; then run the function again with a neighbouring solution and calculate the difference between the two solutions. If the difference is bigger than zero, the new solution will be chosen as a temporary solution and the old one discarded.

If the difference between the two functions is negative, a decreasing probabilistic function will evaluate if the solution value should be updated. This decreasing value simulates the decreasing temperature of the system. In the beginning of the process, there is a big possibility that a bad value will be temporarily chosen, but that helps the algorithm to evade local optimal solutions. As the temperature decreases, the chances of an unwanted value decreases, and the system converges to a global solution (JIANLAN; YUQIANG; XUANZI, 2010).

To apply this algorithm to the routing problem, we have to describe a function that receives a list of target coordinates, organizes it randomly, and then a fitting function calculates the distance it would take to visit all the targets iterating through a distance matrix. Once it has the first distance, it mutates the strategy of visit and calculates fitting again. If the distance is smaller than the previously calculated, this strategy is the new one. If the distance is bigger than the current strategy, it will be then compared to the temperature of the system, simulated by the decreasing probabilistic function, to identify if it will be discarded or assume the new strategy, even if it means a temporary setback into the routing performance.

## 4.4 Immune Algorithm

The Immune Algorithm, (IA) is the last of the bio inspired algorithms used in this project, and, as stated by its name, aims to partially emulate functions of the human body responsible to identify and neutralize antigens. Chu et al (2008) simplifies it’s functioning in nine steps that will be directly compared to the routing problem.

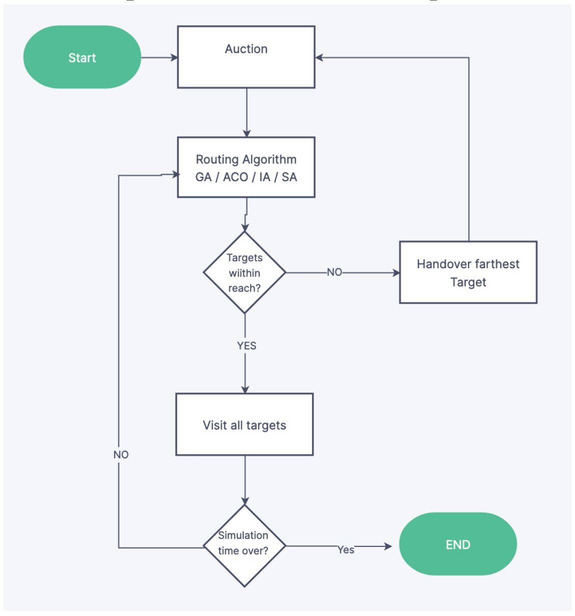
1. Define the antigen, the problem to be solved, and identify what should be the problem's variables and its evaluation function. The problem to be solved is what route to chose when visiting several moving targets, the function variables are the list of targets coordinates and a distance matrix. The goal of the problem is to minimize travelling distance.
2. Generate the initial group of solutions, the antibodies. The first set of antibodies are the first set of routing solutions: randomly generated travelling plans with the order of every target to be visited.
3. Run the evaluation function against the initial antibodies. Using the distance matrix, every antibody will have the total travelling distance of the plan calculated to be compared in the next step.
4. Choose the best antibodies.
5. Clone the set with the best antibodies. The route with the smaller flying distance is chosen.
6. To avoid local optimums, make small changes to the cloned set of antibodies. This can be done by swapping the order of a few targets in the routing strategy.
7. Evaluate the new set.
8. If the new set is deemed a better fit than the actual antibody set, save it. Otherwise, keep the current set.
9. If the termination criteria were not met, loop back to 5. The termination criteria in this case will be a number of iterations defined to make the algorithm

There are several similarities between IA and the previously mentioned GA, but its important to point out the differences: the search objectives in IA can aim for multiple optimal points while GA points to one optimal solution; the operators in GA are the crossovers and mutations while IA can also use clonal selection and concentration adjustment; and while the evolution criteria in a GA is based in one fitness function, the IA, seeking to reproduce a diverse set of possibilities, can integrate other variables and factors to its indicator (WANG; ZHANG; MAN, 2010).

## 4.5 Developed Solution

the solution suggested by Moraes and Freitas (2020) to the crowd monitoring consists in three different systems: an audition system that distributes the targets to be monitored between the UAVs, a route solver that chooses the optimal route to visit the targets and a handover system that removes targets from UAVs once they are deemed too difficult to reach within an acceptable time.

Figure 4.2 – Simplified flowchart of the implemented solution



The auctions may happen in two different moments, first, immediately once the test starts, delivering the initial set of targets for each UAV, and then every time a new target is marked for handover. In the auction, every unmonitored target receives a bid from the UAVs that participate in the competition. The bid is based in the function 4.1:

$$Bid(u,t) = K_a * (\frac{1}{v_t(D_{ut} + K_b \sum_{i=1}^{n_{uq}} D_{ti})n_{uq}}) \tag{4.1}$$

- u represents an UAV.
- t represents a target.
- $K_a$ is a constant value of $1m^2/s$.
- $K_b$ is a constant value of 25.
- $v_t$ is the velocity of the target.
- $D_{ut}$ is the distance between UAV bidding and target.
- $v_t$ is the velocity of the target.
- $D_{ti}$ is the distance between the target auctioned and other targets monitored by the bidding UAV.
- $n_{uq}$ is the number of targets assigned to the bidding UAV.

Following the rules in the above-mentioned equation, the UAV with the highest bid gets the target assigned to its monitoring list. The variables are build to balance the number of targets monitored by each UAV, but also take into account the distance travelled between targets and the speed they move (MORAES; FREITAS, 2020).

Once the first bid is done, the system is ready for the route solver. Each UAV, running on its own different thread, sends the list of targets to the solver and receives the ordered list back, with the total distance the UAV flies to reach all the targets. The distance is to be used to define if the UAV can start the monitoring mission or proceed to handover one of its targets.

The solver uses one of the algorithms described in the Proposed Solution section (AG, ACO, SA, IA), depending on what algorithm is being tested. Just like suggested by De Moraes and De Freitas (2020), the problem can be modelled as a time dependent travelling salesman problem where, after each round, the cost of travelling through the edges changes in time because targets move. It's very important to remember that the travelling salesman is a PN problem, that means a non-deterministic polynomial problem, so for each new target to be visited the complexity rises exponentially (YANG; WANG, 2016), making any optimal solution very costly. The project faces this limitation, deciding an optimal travelling strategy should be chosen in a time close to one second. All its parameters are adapted do this limitations.

The project did not implement its own versions of the routing algorithms, but relied on the python library called Scikit-opt, already optimized for the problem. For GA, ACO and IA, the initial parameters were the ones closest to Moraes and Freitas (2020): Population size 26 and 10 iterations, while for the SA, parameters were varied until found values for maximum temperature (250 degrees), minimum temperature (50 degrees) and chain size (5).

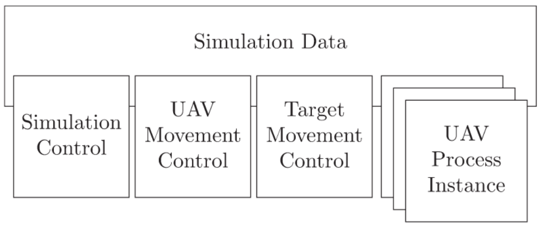
Once the ordered list of targets is delivered to the UAV, it's decided if the round of visits starts or the handover process. The decision is made based on drone max speed (20 m/s), field of view (20 m radius) and target max speed (2 m/s). Ideally, if the drone would take more than 10 seconds (200 m) to reach a target, there is a chance it's already missed.

Unfortunately testing the algorithm, with viable processing time close to one second, very few runs of the route solver reached the 200 m limit, causing many handovers and slowing the general runtime. Moraes and Freitas (2020) suggest also the addition of movement prediction, our implementation adds the target speed times the time that has passed between the start of the mission to adjust the goal of the UAV before it starts moving to the target.

To make sure that missions are actually finding targets, once the UAV reaches the point where it expects the target to be, a test is done. If the distance is smaller than 20 m, it is counted as a visit, if not it's ignored and the UAV moves to the next target. With this addition, it was possible to use a maximum distance of 300 m instead of 200 m.

Another aspect of the test implementation worth mentioning is the multithreading. To keep the systems separated, the target control, simulation control and each UAV runs on different threads. This was implemented using the python standard library thread and semaphores every time a resource needs to be shared.

Figure 4.3 – Threaded simulation environment.



Source: (MORAES; FREITAS, 2020)

One last aspect of the simulation that needs to be addressed is navigation. To keep the project as close as possible to De Moraes and De Freitas (2020), no complex navigation algorithm was used. Since the navigation environment consists in a simple plan with no barriers or complex topology, the basic strategy of movement is to turn the UAV Yaw axis directed to the target, then accelerate to it until reaching a distance smaller than 20 m, then proceed to the next target. This solution is popular in basic Ros-Gazebo tutorials like Tellez (2017).

# 5 SOLUTION IMPLEMENTATION

This project was fully implemented in an open source framework called ROS, the Robot Operating System, a standard development kit for robotics. Its main advantages are a big community building, testing development packages and its multipurpose architecture, where all kinds of robotics can be created.

Suvarna et al. (2019) describes the Robot Operating System as the backbone of a robot simulation, helping the engineer to abstract hardware, handle files, and build projects using languages like C++ and Python. The operating system has a system of nodes and topics that provide an interface to communicate between different components of a project or even different entities in a simulation.

The other software used to simulate the system is Gazebo, an "open source simulator which was developed to effectively and realistically simulate the environment faced by robots in operation" (SUVARNA et al., 2019). In Gazebo it is possible to design the world where the tests will happen, adding models created by the engineer or by the community. Between the features used in this project, Gazebo provides a physics engine, basic sensors and plugins for movement and odometry.

This project used a set of packages called Gazebo-ROS Packages that help integrate the two systems, while the gazebo simulates the world, it's possible to use programming languages to access ROS functions and affect the entities and the environment. The main plugin used was the Planar Move Plugin, used to get odometry values from the UAVs and control its speeds and acceleration. This plugin is useful if the project doesn't need to completely design a full working robot for every single moving object, giving simple moving capabilities to simple entities like cubes and spheres.

This project relies heavily on Moraes and Freitas (2020) to describe the environment and the tests. The goal is to work as close to their approach as possible and compare them, to understand not only how the different algorithms compare but also how the projects, in different environments, with their own implementations and simplifications can compare to each other.

Moraes and Freitas (2020) built a custom ad-hoc solution to simulate the environment for a reason: "At high speeds and strict temporal constraints, even the slightest delay in processing some information or making some decision can heavily influence the activity outcome, sometimes crossing the line between success and failure." While their work already shows that the solution is possible, this work implements it in an open platform.

This work used Ros-Gazebo, installed in an Ubuntu 20.04.4 LTS virtual machine, to implement 12 different scenarios that tested the behaviour and the scalability of the proposed solution. The implementation started with the environment, a simple Cartesian plane, where a Gazebo world was created following the rules of the test, containing the respective number of targets for every test and the five UAVs.

Table 5.1 – Basic Simulation Parameters

| | |
|---|---|
| Simulation time | 900s |
| Simulation area | 500 x 500 m |
| Number of UAVs | 5 |
| Number of targets | 25 |
| Target initial speed | random from 0 to 2 m/s |
| Target initial heading | random |
| Target speed variation | random from 0 to 0.5 changing every 20s. |
| UAV speed | 20 m/s |

Source: (MORAES; FREITAS, 2020)

To restrain world size to 500 m x 500 m, instead of building walls and implementing obstacle avoidance, there is a test for every time a target starts moving, if the final destination will be outside the limits, its speed direction changes, pointing the movement in the opposite direction. Since the targets are supposed to move at almost fixed speeds, depending on the test, this is enough to make them move within limits. Unless the target is supposed to hit the world's boundaries, it walks at a constant speed inferior to 2 m/s set randomly in the beginning of every simulation.

The UAV is specified after the 3DR Iris+ quad-copter, the same used in De Moraes and De Freitas (2020), with maximum speed of 20 m/s, lasting batteries of 16 to 22 minutes. The field of view is defined as 20 m radius, equivalent to a Raspberry Pi Sony camera at 45 m high.

# 6 SIMULATION RESULTS AND ANALYSIS

A total of 12 tests were run for every different routing solution, the first six evaluated the correct behaviour of the system, varying the use of movement prediction, handover and speed changes with 25 targets. The scenarios numbered 7-11 evaluate how the system reacts to the rise on the number of targets, from 20 to 100. The final test simulates a crowd moving longitudinally in a long street, just like demonstrators would in a narrow venue. All tests follow the designs of De Freitas and De Moraes (2020).

The number of tests were chosen to aim for an error of 5% of the mean or less, in a 95% confidence interval. This standard error was found through the following equation: $E = Z(c) * \frac{\sigma}{\sqrt{n}}$ where Z(c) is the standard score used in a normal distribution (1.96) for $95\%$ confidence, $\sigma$ is the standard deviation and n is the size of the sample. The project aimed for error below $5\%$.

## 6.1 Behaviour Tests

This set of tests is designed to verify if the designed system is working as expected. The parameters of the tests are the same as in table 7.1. The first scenario runs the operation with no handover or prediction, the second turns on the prediction, the third works with the handover but no prediction, the fourth has both features turned on. The last two tests of this set activate the speed variation every 20 seconds in the targets. This is summarized in the table below:

Table 6.1 – Behaviour Simulation Setup

| **Scenario** | **Handover** | **Movement Prediction** | **Speed** |
|---|---|---|---|
| Scenario 1 | Disabled | Disabled | Disabled |
| Scenario 2 | Disabled | Enabled | Disabled |
| Scenario 3 | Enabled | Disabled | Disabled |
| Scenario 4 | Enabled | Enabled | Disabled |
| Scenario 5 | Enabled | Disabled | Enabled |
| Scenario 6 | Enabled | Enabled | Enabled |

Source: adapted from (MORAES; FREITAS, 2020)

The table 6.2 shows the results of the simulations implemented by Moraes and Freitas (2020). The goal here is to observe not the absolute numbers, but if the patters of behaviour confirm. The tendencies we wish to observe improvements on scenarios two and three, compared to the first and the fourth scenario should have the best performance. Scenarios five and six should show some deterioration against the fourth, caused by the increased chaos caused by more constant changes of speed and velocity.

Table 6.2 – The first scenario has handover and movement prediciton disabled

| Scenario | Visit Count | Average time between visits |
|---|---|---|
| Scenario 1 | 26 | 33.7 |
| Scenario 2 | 33.6 | 27.8 |
| Scenario 3 | 31.5 | 29.1 |
| Scenario 4 | 35.9 | 25.4 |
| Scenario 5 | 23.9 | 36.7 |
| Scenario 6 | 24.4 | 34.2 |

Source: (MORAES; FREITAS, 2020)

The table 6.3 shows the results of the first series tests. The variable column differentiates two measures, the average time between visits (measured in seconds) in one test and the average number of visits each target receives in one test. The algorithm is one of the four routing algorithms measured, and the error is the percent of error compared to the result.

The first scenario unveils expected differences in absolute numbers, but the general tendencies remain. Each of the six different behaviour tests show us the system variables in a different situation.

Table 6.3 – The first scenario has handover and movement prediciton disabled

| Variable | Algorithm | Result | Error |
|---|---|---|---|
| Time between visits | ACO | 129.23 | 4.7% |
| Number of Visits | ACO | 3.61 | 9.0% |
| Time between visits | GA | 124.44 | 5.6% |
| Number of Visits | GA | 3.96 | 9.7% |
| Time between visits | IA | 115.52 | 4.8% |
| Number of Visits | IA | 3.26 | 9.4% |
| Time between visits | SA | 127.12 | 4.0% |
| Number of Visits | SA | 3.74 | 8.6% |

Source: Behaviour Tests

The first test, with both handover and movement prediction disabled, is supposed to have the worst numbers in this set, just like in the benchmark work. The poor results in table 7.2 show that, most of the time, the agent is not finding the target at the expected place, this happens because the trip is taking too long and the target already left the 20 m surveillance radius. The best performance was the Genetic Algorithm, but as we can see through the tests, the difference between algorithms usually is smaller than the average error.

Table 6.4 – The second scenario has handover disabled and movement prediction enabled

| **Variable** | **Algorithm** | **Result** | **Error** |
|---|---|---|---|
| Time between visits | ACO | 82.64 | 4.7% |
| Number of Visits | ACO | 12.26 | 4.7% |
| Time between visits | GA | 76.63 | 5.1% |
| Number of Visits | GA | 14.12 | 4.3% |
| Time between visits | IA | 76.44 | 3.0% |
| Number of Visits | IA | 13.6 | 3.8% |
| Time between visits | SA | 77.47 | 3.8% |
| Number of Visits | SA | 13.8 | 5.2% |

Source: Behaviour Tests

The second scenario, table 6.4, shows some significant improvements. The Simulated Annealing shows the best results, reaching close to half of the benchmark results. If we consider the maximum error of 9.7% there are still draws between the algorithms implemented here. The important hint in the data here is how significantly movement prediction can improve the results.

Table 6.5 – The third scenario has handover enabled and movement prediction disabled

| **Variable** | **Algorithm** | **Result** | **Error** |
|---|---|---|---|
| Time between visits | ACO | 121.15 | 5.6% |
| Number of Visits | ACO | 4.59 | 9.4% |
| Time between visits | GA | 128.72 | 6.6% |
| Number of Visits | GA | 4.72 | 9.0% |
| Time between visits | IA | 117.33 | 5.3% |
| Number of Visits | IA | 4.58 | 9.8% |
| Time between visits | SA | 125.39 | 3.9% |
| Number of Visits | SA | 5.34 | 9.7% |

Source: Behaviour Tests

The third scenario (table 6.5 )hits numbers just as bad as the first one, this can be explained for the same reason. Even with the handover process, there are a lot of misses when the UAV arrives at the target.

The important scenario is the fourth one (table 6.6), that significantly reduces the time between the visits to less than a minute per hit. The GA outperforms the other

Table 6.6 – The fourth scenario has handover and movement prediction enabled

| **Variable** | **Algorithm** | **Result** | **Error** |
|---|---|---|---|
| Variable | Algorithm | Result | Error |
| Average Time | ACO | 65.26 | 3.9% |
| Average Visits | ACO | 13.83 | 4.8% |
| Average Time | GA | 61.06 | 4.0% |
| Average Visits | GA | 14.78 | 5.0% |
| Average Time | IA | 65.58 | 3.8% |
| Average Visits | IA | 14.15 | 5.3% |
| Average Time | SA | 62.62 | 3.4% |
| Average Visits | SA | 14.33 | 6.2% |

Source: Behaviour Tests

algorithms by a very small amount, still having a draw in the error margin. In this case, each target will be visited each 61.06 seconds in average.

Table 6.7 – The fifth scenario has speed changes and handover enabled

| **Variable** | **Algorithm** | **Result** | **Error** |
|---|---|---|---|
| Time between visits | ACO | 112.94 | 6.2% |
| Number of Visits | ACO | 3.48 | 9.9% |
| Time between visits | GA | 99.2 | 6.0% |
| Number of Visits | GA | 3.97 | 9.9% |
| Time between visits | IA | 103.09 | 5.3% |
| Number of Visits | IA | 3.7 | 9.5% |
| Time between visits | SA | 91.45 | 8.7% |
| Number of Visits | SA | 3.89 | 9.4% |

Source: Behaviour Tests

The scenarios five and six (6.7 and 6.8) should be compared against scenarios three and four. In the scenarios with movement prediction and handover, it could take an extra 18 seconds in average to visit a target. This shows how adding more speed and variability to the target's behaviour, or something like a more violent crowd, will make monitoring more difficult.

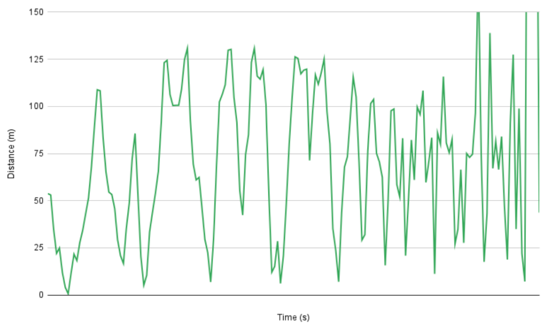
The figure 6.1 is designed to better visualize the behaviour changes between target and UAV . One random target was chosen from the fourth scenario, the one with handover and prediction enabled. This target was monitored, measuring the smaller distance between all the UAVs running the test. This case also indicates some kind of degradation at the end of the test, where distances between UAV and target start to get bigger than 150 m.

Table 6.8 – The sixth scenario has handover, movement prediction and speed variation enabled

| Variable | Algorithm | Result | Error |
|---|---|---|---|
| Time between visits | ACO | 84.15 | 4.6% |
| Number of Visits | ACO | 9.41 | 6.5% |
| Time between visits | GA | 81.32 | 4.7% |
| Number of Visits | GA | 10.23 | 6.6% |
| Time between visits | IA | 80.93 | 4.5% |
| Number of Visits | IA | 10.48 | 5.2% |
| Time between visits | SA | 81.21 | 4.4% |
| Number of Visits | SA | 10.12 | 5.9% |

Source: Behaviour Tests

Figure 6.1 – Distance between target and closest UAV in one full scenario



Source: Behaviour test data from the Scenario 4 - Algorithm: GA -Time: 900s.

## 6.2 Scalability Test

The scalability analysis seeks to identify how well the system handles the adding of extra targets to the environment. The first test has 20 targets, four for each UAV, the next ones 30, 40, 60, 100. Handover and movement prediction were enabled for all the test. The basic information of the scenario and the original work results using a GA are below on table 6.9:

Table 6.9 – Scalability simulation setup and benchmark results

| Scenario | targets | Handover | M. Prediction | Speed | Visit Count | Visit Time |
|---|---|---|---|---|---|---|
| 7 | 20 | Enabled | Enabled | Constant | 36.5 | 24.97 |
| 8 | 30 | Enabled | Enabled | Constant | 30.6 | 29.41 |
| 9 | 40 | Enabled | Enabled | Constant | 29.15 | 30.88 |
| 10 | 60 | Enabled | Enabled | Constant | 30 | 29.99 |
| 11 | 100 | Enabled | Enabled | Constant | 30.03 | 29.95 |

Source: Adapted from Moraes and Freitas (MORAES; FREITAS, 2020)

The significant pattern to observe in the original work is the expected reduction of performance while the number of targets rise. What is maybe unexpected is the saturation of the degradation once the number of targets hit 60. Moraes and Freitas (2020) point that more targets in the area may reduce the distance between them, leading to stable performances even if the number of targets rise.

Table 6.10 – Scalability test with 20 targets

| Targets | Variable | Algorithm | Result | Error |
|---|---|---|---|---|
| 20 | Number of Visits | ACO | 17.75 | 4.4% |
| 20 | Time between Visits | ACO | 46.36 | 5.7% |
| 20 | Number of Visits | GA | 17.81 | 3.1% |
| 20 | Time between Visits | GA | 45.95 | 4.4% |
| 20 | Number of Visits | IA | 18.05 | 3.2% |
| 20 | Time between Visits | IA | 48.1 | 4.4% |
| 20 | Number of Visits | SA | 18.69 | 3.7% |
| 20 | Time between Visits | SA | 47 | 4.0% |

Source: Scalability tests

This test follows some of the Moraes and Freitas (2020) trends of results, in a way that, like expected, fewer targets will result in better performance. The test with 20 targets had the best numbers, with the SA algorithm hitting a total of 19.01 times each target, in average, and taking 46 seconds to return between visits.

One trend that the study's results don't follow is the stabilization of performance after 30 cases, unlike De Moraes and De Freitas (2020): where the results from scenarios with 60 and 100 targets clearly stabilize, this work's scalability tests shows notable degradation of the performance, with the cost of adding targets raising more on every new target added.

Table 6.11 – Scalability test with 30 targets

| Targets | Variable | Algorithm | Result | Error |
|---|---|---|---|---|
| 30 | Number of Visits | ACO | 11.3 | 3.0% |
| 30 | Time between Visits | ACO | 71 | 4.5% |
| 30 | Number of Visits | GA | 10.77 | 4.3% |
| 30 | Time between Visits | GA | 59.22 | 3.4% |
| 30 | Number of Visits | IA | 11.58 | 3.0% |
| 30 | Time between Visits | IA | 69.47 | 5.5% |
| 30 | Number of Visits | SA | 11.64 | 3.1% |
| 30 | Time between Visits | SA | 71.13 | 3.4% |

Source: Scalability tests

The data on the scalability tests also shows the significant change in time between the scenarios, and another interesting situation: even though within the margin of error, the ACO algorithm starts to show some advantages to the others when the number of targets rise. This study still can't conclude this is the best algorithm because at a 4.5% error, it's still a draw with SA in the last scenario.

Table 6.12 – Scalability test with 40 targets

| Targets | Variable | Algorithm | Result | Error |
|---|---|---|---|---|
| 40 | Number of Visits | ACO | 8.36 | 3.3% |
| 40 | Time between Visits | ACO | 83.26 | 3.8% |
| 40 | Number of Visits | GA | 8.54 | 2.7% |
| 40 | Time between Visits | GA | 83.02 | 5.2% |
| 40 | Number of Visits | IA | 8.52 | 2.9% |
| 40 | Time between Visits | IA | 86.51 | 6.5% |
| 40 | Number of Visits | SA | 8.62 | 3.3% |
| 40 | Time between Visits | SA | 82.95 | 5.3% |

Source: Scalability tests

Table 6.13 – Scalability test with 60 targets

| Targets | Variable | Algorithm | Result | Error |
|---|---|---|---|---|
| 100 | Number of Visits | ACO | 2.7 | 4.7% |
| 100 | Time between Visits | ACO | 98.71 | 5.6% |
| 100 | Number of Visits | GA | 3.21 | 2.4% |
| 100 | Time between Visits | GA | 171.32 | 4.4% |
| 100 | Number of Visits | IA | 3.15 | 3.1% |
| 100 | Time between Visits | IA | 175.31 | 3.3% |
| 100 | Number of Visits | SA | 3.4 | 2.3% |
| 100 | Time between Visits | SA | 150.8 | 2.2% |

Source: Scalability tests

## 6.3 Experiment Considering Ordinary Urban Walking Scenario

The last experiment of this study aims for reproducing an environment as close as possible to a street demonstration where 25 targets walk longitudinally in a long avenue (1500 x 50 m) for 900 seconds. Every target moving on its own speed between 0 and 2 m/s. This test was build to verify if a more concentrated environment could result in better results for crowd monitoring.

Table 6.14 – Scalability test with 100 targets

| Targets | Variable | Algorithm | Result | Error |
|---|---|---|---|---|
| 30 | Number of Visits | ACO | 11.3 | 3.0% |
| 30 | Time between Visits | ACO | 71 | 4.5% |
| 30 | Number of Visits | GA | 10.77 | 4.3% |
| 30 | Time between Visits | GA | 59.22 | 3.4% |
| 30 | Number of Visits | IA | 11.58 | 3.0% |
| 30 | Time between Visits | IA | 69.47 | 5.5% |
| 30 | Number of Visits | SA | 11.64 | 3.1% |
| 30 | Time between Visits | SA | 71.13 | 3.4% |

Source: Scalability tests

Table 6.15 – Urban Scenario Setup

| Targets | Handover | M. Prediction | Speed | Simulation area |
|---|---|---|---|---|
| 25 | Enabled | Enabled | Constant | 1500 x 50 m |

Source: Adapted from Moraes and Freitas (MORAES; FREITAS, 2020)

Once again, the benchmark study and this work disagree that a more compact crowd would result in better monitoring performance. Different from Moraes and Freitas (2020), a higher concentration of target in a smaller area does not mean improved performance, pointing that other factors than only distance between target and UAV may influence in the final results.

Table 6.16 – Simmulation of urban scenario

| Variable | Algorithm | Results | Error |
|---|---|---|---|
| Number of Visits | GA | 14.22 | 4.6% |
| Time between Visits | GA | 58.2 | 3.6% |
| Number of Visits | ACO | 13.31 | 4.4% |
| Time between Visits | ACO | 62.86 | 3.5% |
| Number of Visits | SA | 13.67 | 2.7% |
| Time between Visits | SA | 66.42 | 4.4% |
| Number of Visits | IA | 14.69 | 4.7% |
| Time between Visits | IA | 58.17 | 3.8% |

Source: Urban Scenario Simulation

Other interesting result is the fact that, once again, no algorithm can be considered an unmatched winner pointing that, within that implementation, makes a significant difference in the final results.

## 6.4 Discussion of Results

The first thing final test results show us that, in this implementation, with a 95% confidence interval, with the maximum error of the tests being around 5% is that no algorithm can be trusted to best the others. In spite of the final average of the IA being the smaller value, it's still under the 4.4% margin of error.

This first analysis of the data signs that the routing algorithm may not be a bottleneck in this system, and future research might have to look into other parts of the system to seek relevant improvement. This extends to all the other tests, while sometimes one algorithm can win by a small margin, generally the numbers approximate to a draw.

Another clear result is the very significant difference between the De Moraes and De Freitas (2020) implementation and this work. There are several variables that can explain the negative differences and additional research will be needed to pinpoint exactly what caused the biggest contrast, but implementing both systems in physics engines that evaluate UAV linear acceleration, changes in direction, air resistance and inertia can cause attrition to the numbers. Other possible variables are UAV navigation and control, not fully described in De Moraes and Freitas (2020) and may cause big differences depending on the decisions made.

That being said, the patterns remain comparable. The behaviour test very similar in the compared results, and other tests have some differences that can be explained by differences in implementation and counting of positives.

One extra reason for the numbers difference is that a visit in this work is only really counted if one UAV leaves its original point and find the target to be in a distance smaller than 20 m when it reaches the location, not counting any target that it might have past accidentally around, causing considerable differences on patterns in the scalability analysis. It doesn't matter how full is the environment, the UAV only hits a target it is looking for.

The data shows that this implementation does not achieve the necessary performance to conduct full surveillance in all the targets during the test. Using Moraes and Freitas (2020) metric of success indicates that a pedestrian will move at speed of 1.25 m/s, with maximum deviation of 0.5 m. In the worst case it would take 20 seconds for a walker to evade surveillance, a total of 70,8% of the targets. In the average case, 30 seconds for evading, the project would still lose more than half of the targets (56,2%).

There is one important positive point to look into. The improvement of 52 % of the results comparing scenarios where the movement prediction is enabled may hint for a bottleneck to be explored. A deeper research in human behaviour in crowds for the creation of more precise predictors could be explored.

# 7 CONCLUSIONS

This work successfully simulated the problem of monitoring a crowd with multiple UAVs testing four different swarm intelligence algorithms to identify the optimal performance. The work also successful compared with the Moraes and Freitas (2020) implementation, following some of the same patterns on the behaviour and scalability tests, and its differences have been explained. Also, the development and testing of the project in an open source platform is an invitation for others to build on top of what was already developed.

Immediate future works in this platform should focus on reducing time between visits by improving on different aspects than which algorithm to run to choose the routing. Another possibility is testing the performance against a more efficient programming language like C++. One can also improve on navigation and control, deepening the use of other important robotics concepts like simultaneous localization and mapping and implementing the platform in hardware to see real life results. There is also the possibility of adding heterogeneous systems where different UAVs collaborate in different activities for the success of a mission.

In a different aspect, actually verifying with local law enforcement agencies their needs and resources and adapting the project to them could bring interesting inputs from professionals that deal with the problem of crowd monitoring on a weekly basis.

It's also important to add that this work was developed as a final graduation project for a computer engineering bachelor. It helped to test and improve skills developed during the whole course, like statistics, data structures, programming, robotics, data architecture, artificial intelligence and scientific research.

## REFERENCES

DORRI, A.; KANHERE, S. S.; JURDAK, R. Multi-agent systems: A survey. **IEEE Access**, v. 6, p. 28573–28593, 2018.

ERGEZER, H.; LEBLEBICIOGLU, K. Path planning for uavs for maximum information collection. **IEEE Transactions on Aerospace and Electronic Systems**, v. 49, n. 1, p. 502–520, 2013.

FREITAS, E. P. de et al. Decentralized task distribution among cooperative uavs in surveillance systems applications. In: **2010 Seventh International Conference on Wireless On-demand Network Systems and Services (WONS)**. [S.l.: s.n.], 2010. p. 121–128.

GARCIA, E. et al. The evolution of robotics research. **IEEE Robotics Automation Magazine**, v. 14, n. 1, p. 90–103, 2007.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. Ann Arbor, MI: University of Michigan Press, 1975. Second edition, 1992.

JIANLAN, G.; YUQIANG, C.; XUANZI, H. Implementation and improvement of simulated annealing algorithm in neural net. In: **2010 International Conference on Computational Intelligence and Security**. [S.l.: s.n.], 2010. p. 519–522.

KHAN, A.; RINNER, B.; CAVALLARO, A. Cooperative robots to observe moving targets: Review. **IEEE Transactions on Cybernetics**, v. 48, n. 1, p. 187–198, 2018.

MORAES, R.; FREITAS, E. Pignaton de. Distributed control for groups of unmanned aerial vehicles performing surveillance missions and providing relay communication network services. **Journal of Intelligent Robotic Systems**, v. 92, 12 2018.

MORAES, R. S. de; FREITAS, E. P. de. Multi-uav based crowd monitoring system. **IEEE Transactions on Aerospace and Electronic Systems**, v. 56, n. 2, p. 1332–1345, 2020.

SUVARNA, S. et al. Simulation of autonomous airship on ros-gazebo framework. In: **2019 Fifth Indian Control Conference (ICC)**. [S.l.: s.n.], 2019. p. 237–241.

TIWARI, K. **Multi-robot exploration for environmental monitoring : the resource constrained perspective**. Amsterdam, Netherlands: Academic Press, 2020. ISBN 0-12-817608-3.

WANG, F.; ZHANG, D.; MAN, L. Comparison of immune and genetic algorithms for parameter optimization of plate color recognition. In: **2010 IEEE International Conference on Progress in Informatics and Computing**. [S.l.: s.n.], 2010. v. 1, p. 94–98.

YANG, X.; WANG, J.-s. Application of improved ant colony optimization algorithm on traveling salesman problem. In: **2016 Chinese Control and Decision Conference (CCDC)**. [S.l.: s.n.], 2016. p. 2156–2160.