Politecnico di Torino

Department of Mechanical and Aerospace Engineering

Development of a multi-UAS coverage planning algorithm based on the Ant Colony optimization

Master Degree in Aerospace Engineering

Supervisor:

Prof. Giorgio Guglieri

Co-Supervisor:

Dr. Stefano Primatesta

Candidate:

Misino Francesco

288695

A.A. 2021-2022

Summary

This thesis focuses on the coverage planning problem with multiple UAVs (Unmanned aerial Vehicles), also called drone. The aim is to minimize the coverage time of an operational area with the presence of several obstacles. The strategy adopted divides the operational area to be covered into several regions, and, then, assigns them to the corresponding UAV.

The choice for the optimization algorithm was made through a literature review of the different methods proposed at the state of the art, focusing on those peculiar to multi-drone aerial systems (Chapter 2). The method adopted is based on the use of Voronoi diagram and Delaunay triangulation to decompose the operational area into regions. Thus, an algorithm based on the bio-inspired Ant Colony method provides an optimization by assigning the different regions to each UAVs, as well as defining the order of visit each region, evaluating the parameters of drones, such as cruise speed and the field of view of the sensor used in the coverage application.

For the subdivision of the area, two methods were mainly investigated. First, the Boustrophedon decomposition has been used, which involves the subdivision into regions based on the presence of a obstacles in the operational area. Second, the Voronoi diagram is exploited by generating points (seeds) of the regions using a Conforming Constrained Delaunay Triangulation of the whole operations area (Chapter 3). The resulting behavior of this strategy depends on the use of constraining parameters linked to the area definition for the correct tessellation. Several tests have been performed to select the best approach for the decomposition of the area, but the Voronoi-Delaunay combination allows a faster coverage and a more equitable division into regions and, then, assignment to the various drones.

For the region assignment and order of visiting, the Ant Colony has been adopted (Chapter 4). It is a bio-inspired approach based on the natural behavior of ants that choose the best

path between two destinations exploiting both the previous knowledge of the environment and the presence of pheromone left by the ants themselves. In fact, it is observed that the selected paths are gradually optimized by selecting the shortest one. This strategy is widely adopted by the state of the art for the path planning of autonomous vehicles, e.g. drones in this thesis. Specifically, the method involves the use of two parameters, heuristic and pheromonic, that vary between the regions, until the solution converges to the best one between the different artificial ants.

The proposed coverage planning strategy has been implemented using Python, used to generate the operational scenario (i.e. the map to be covered), decompose the operation area to regions and, then, assign the regions to drones that visit the assigned regions. The obtained results of the optimization demonstrate both the usefulness of the pheromone information for the ant colony algorithm and the best choice through the subdivision with Voronoi diagram (Chapter 5).

Contents

Chapter 1

Introduction

The growing use of robots and remotely piloted vehicles has introduced the need for the development of intelligent control and planning algorithms . In particular, unmanned and autonomous systems are used to provide several applications such as monitoring, mapping, surveillance, to name a few. The use of an unmanned system permits to monitor both indoor and outdoor environments without the involvement of humans and, in case of high risk operations, avoiding subjecting humans to excessive risks. For this purpose, several unmanned systems have been involved in these applications as unmanned ground vehicles (UGV), unmanned air vehicle (UAV), and autonomous underwater vehicles (AUV).

The possibilities of using these systems, especially for air vehicle, are many, such as surveillance, sampling through sensors, mapping of overflown areas, smart farming, fire tracking, photogrammetry, civil security, management of environmental disasters and others.

There are several issues concerning the coverage of autonomous robots and in this thesis project the focus is on UAVs, being a theme for the degree course in question; these, resume the work initially carried out for the ground robots, have characteristics common to the latter, however they allow the coverage of areas faster, with fewer constraints on terrestrial obstacles, but, on the other hand, having poorer energy performance, with lower autonomy due to the reduced weights.

The coverage of the areas is mainly divided into two steps:

- Generation of viewpoints

- Development of path planning

Furthermore, these two topics are influenced by the type of air vehicle chosen, with related kinematic constraints and the type of scenario to be evaluated, with variable geometries, more or less extensive areas and the possibility of an evolution of this over time. Moreover, in the case of very large areas, it is possible to use a system of several robots (multi-UAV system), which allows coverage in less time, however introducing problems concerning cooperation and information sharing between them.

In the following Chapter, the two main topics are dealt with a focus on multi-UAV systems.

Outline

The thesis work consists of a first review of the state of the art, in Chapter 2, a subsequent analysis of the two methods chosen for the decomposition of the operational area, Boustrophedon Cell Decomposition and Voronoi diagram, in Chapter 3, a broad explanation of the Ant Colony method used for the allocation and ordering of cell visits in Chapter 4 and, finally, the presentation of the algorithm implementation on Python and of the Results in the Chapter 5; in the last Chapter conclusions are drawn and possible future developments of the thesis work are described.

Chapter 2

State of Art

This Chapter presents the topic of Coverage Path Planning with a careful analysis of the state of the art and a subsequent focus on multi drone systems. Finally, it defines the methods selected for the subsequent development of the algorithm.

2.1 Coverage Path Planning

Coverage Path Planning (CPP) is one of the themes that most benefits from the use of robots, and concerns a series of uses and future developments. As for UAVs, it should initially be distinguished for aircraft with rotary-wing (RW) and fixed wings (FW), in fact the former do not require relative speed for sustenance, therefore they also allow hovering flight (as well as helicopters), vertical take-off (VTOL), and in general even broken line trajectories, while the latter cannot be stopped due to the generation of the lift that allows motion, introducing a kinematic constraint in the trajectories that cannot have too small radius of curvature. In addition, fixed-wing aircraft need a landing and take-off space, and must constantly move during the mission profile, on the other hand they allow higher speeds than multi-rotor aircraft and the aerodynamic study is simpler in the absence of interaction phenomena between rotating blades (multi-rotor wings).

Another important distinction concerns the type of information that is possessed and its possible flow through ground systems or multiple systems in flight. In fact, depending on whether they are all prior known, or only partially, the coverage of the area can be carried

out offline in the first case, or online, with the communication of the data available through sensors placed on the UAVs. In the first case, the development of the path can be generated before the flight and does not undergo variations, however the scenario can be dynamic, unforeseen failures can be introduced and, especially in the case of a multi-UAV coverage, it is necessary to foresee the interaction between the different subjects, in relation to what is the evolution of the area, reorienting the path according to the eventualities that developed during the mission.

Once the type of aircraft and the type of planning have been defined, it is possible to focus on the two steps presented above, also with variable peculiarities depending on the scenarios.

2.1.1 Viewpoints Generation

Viewpoints generation defines the positions and orientations of the sensor from where the data will be collected, thus affecting the overall coverage. They can be considered as stages through which the vehicle must pass, in order to collect data and allow the best coverage of the area. Their generation depends on the type of information possessed, therefore they can be prior defined, depending for example on the priority of some areas or the need for maximum coverage, or they can be initially set randomly and subsequently updated according to the information obtained during the flight. The choice can therefore be model or non-model based.

- *Model-based:* having a CPP model-based algorithm allows to have a simpler process in the generation of viewpoints, being able to choose, especially for critical areas such as in disaster scenarios (for example earthquake zones), to give greater weight to certain areas. However, there is no model for everything that can be encountered. One of the most used are grid-based models, with the subdivision into cells of the territory, with a typically triangular, square or hexagonal shape, whose center corresponds to the viewpoints. The work presented by Nedjati A. et al. [1] performs grid based decomposition according to each UAV footprint considering sensor size, focal length, and UAV altitude. The territory is divided into cells that correspond to residential areas, uninhabited areas and energy deposits of an earthquake territory.

- *Non model-based:* the position of viewpoints in non-model based algorithms is not predetermined and the main objective is to maximize coverage by minimizing the energy consumed by UAVs. In this case, the update occurs during coverage, starting from a random allocation. From Vemprala S. and Saripalli S [2], a method of generating viewpoints for a multi-robot coverage is presented, with the sharing of information for robots in line of sight, with the aim of minimizing the path of the single aircraft.

2.2 Path Planning

The topic of path planning deserves a greater discussion as it represents a fundamental point of the CPP, with numerous differences in the various models and different advantages and disadvantages.

The path planning consists in the generation of the UAV route that allows to cover the entire desired area, considering the kinematic and environmental constraints, i.e. the possible maneuvers for the chosen aircraft, or, above all, any obstacles present that are similar to no-flight zones (NFZ), such as buildings, trees, airport areas, etc.

Typical performance functions in this case are total travelled distance or the path length, the time-to-complete a mission, the area coverage maximization, and the number of turning maneuvers, moreover, especially for UAV, also the energy consumption.

For the definition of the problem, the coverage area is identified through a sequence of p_i vertices $\{v_1, ..., v_p\}$, each with coordinates in the plane $\{v_x(i), v_y(i)\}$. Every v_i and v_{i+1} are extremes of the edges $\{e_1, ..., e_i\}$, whose length is equal to $l_i = \|v_i - v_{i+1}\|$. Additionally, the area may contain no-fly zones with vertices $\{u_1, ..., u_p\}$ like in Fig. 2.1.

For multi-UAV systems, in the literature there are different methodologies for generating the path, all with the common goal of avoiding collisions and achieving the coverage of the entire desired area, but different advantages and disadvantages (Tab. 2.2):

- Grid Based;
- Geometric Based;

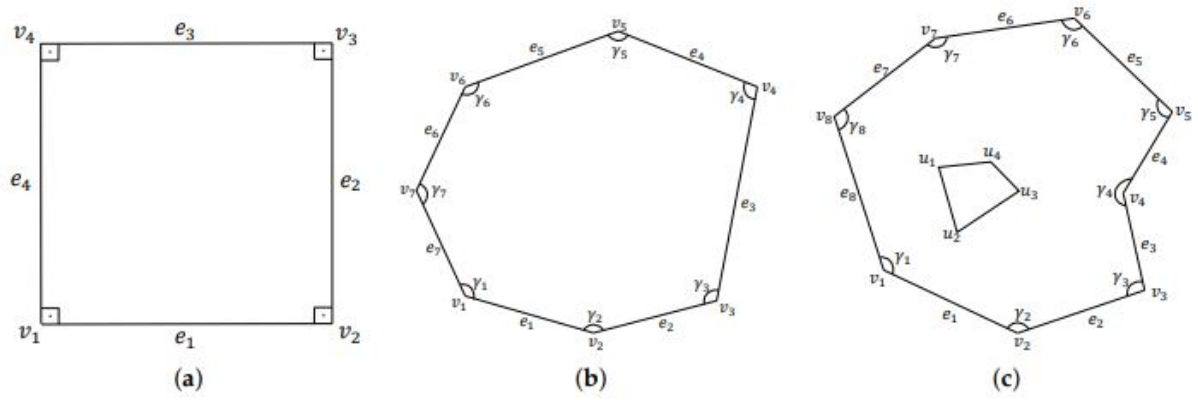


Figure 2.1: Different areas of interest explored during CPP missions: (a) Rectangular; (b) Convex Polygon; (c) Concave Polygon with No-Fly Zones. [3]

- Reward Based;
- New Best View Based.

2.2.1 Grid-Based Path

The Grid-Based approach is actually transversal to numerous applications, being exploited for all the other methodologies with often hybrid solutions. It is a method applicable also for single-robots with the characteristics subsequently proposed, however it is extended for more UAVs with peculiar additions in the model.

According to Choset [4], classical taxonomy presents three possible approaches, with the introduction of the concept of cellular decomposition (subsequently presented). The choice of one approach over another depends mainly on the geometry of the area to be covered and on the type of UAV chosen (Tab. 2.1):

No Decomposition

In reality it is the absence of the use of the aforementioned method, it is possible to use it in the case of very simple geometries, typically convex polygons, and foresees two main types of path, back-and-forth (BF) and the spiral (SP).

The former is based on parallel trajectories with reversal of motion at the end of the stroke

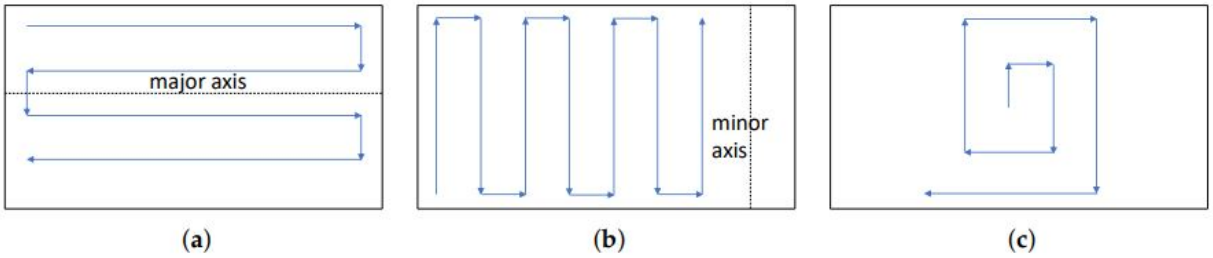


Figure 2.2: Simple flight patterns in rectangular areas with no decomposition: (a) Parallel; (b) Creeping Line; (c) Spiral. [3]

(Fig. 2.2a-b), like an animal pushing a plow in the field, while the latter plans to travel the trajectory parallel to the sides of the polygon (Fig. 2.2c), with 90 degree curves, gradually reducing the radius of the path; for fixed wing vehicle, the turns are softer, while for RW they can also has right angles.

An interesting study on this subject, by Di Franco and Buttazzo [5], presents an energy-aware back-and-forth CPP approach (E-BF) for photogrammetry with energy and resolution constraints imposed by the mission. This approach proposes to minimize the number of turns, selecting the vertex of the longest side of the polygon as the starting point, imposing the scanning direction parallel to this side (Fig. 2.3), and subsequently tracing the path to to satisfy the constraints of sensor's resolution.

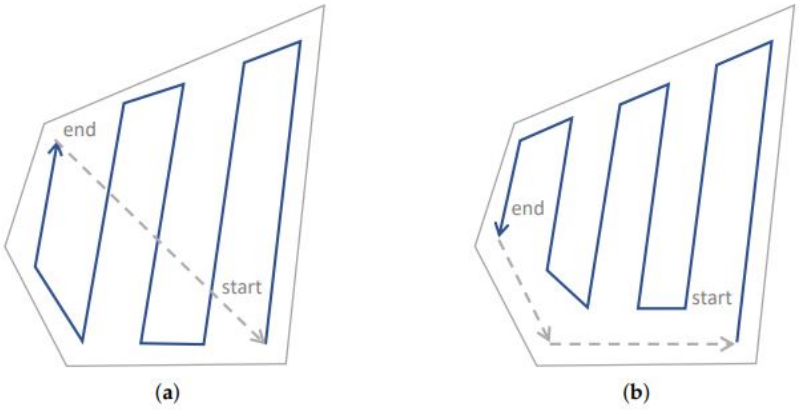


Figure 2.3: Energy-aware back-and-forth coverage path planning algorithm: (a) Odd number of stripes; (b) Even number of stripes. [5]

At the end, the final point and the initial point are connected with a straight path, however

the return path can also be used as a scanning path (Fig. 2.3b). The proposed solution also includes two offline and online failsafe measures, the first conducted before the flight to verify that the path is sustainable from the point of view of the battery, the second during the flight to check the remaining battery necessary for the return to the starting point (hypothetical ground operator or ground station).

Another study, which goes beyond the analysis conducted by Di Franco and Buttazzo, is that of Cabreira et al. [6], which introduces the use of an E-Spiral, with an initial path concentric to the polygon and a subsequent decrease in the radius of the trajectory . This work focuses on the type of turns, which, in general, involve deceleration, attitude change and acceleration, resulting in greater consumption for the system, but the use of wider turns for Cabreira (Fig. 2.4), allows to overcomes the E-BF. Furthermore, both simulations were conducted on real flights, and can be considered the most efficient CPP approach for convex polygonal areas considering energy spent during missions.

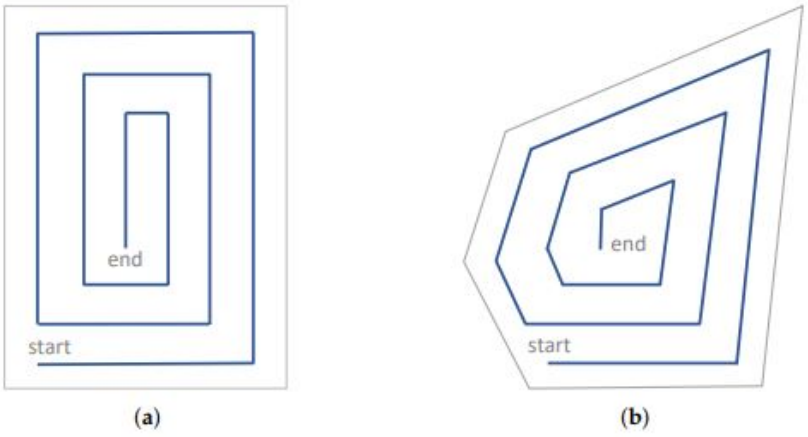


Figure 2.4: Energy-aware spiral algorithm with energy and resolution constraints: (a) Rectangular area; (b) Polygonal area. [6]

Exact Cellular Decomposition

It is the most used CPP method to guarantee total coverage of the desired area, dividing it into smaller and simpler areas of interest. The zones into which the area is divided are typically called *cells*, and bringing them together gives the total area. After dividing the

entire scenario, the problem is simplified to motion within the single cell, performed for example with the type of path presented in the previous section (no decomposition), with the addition of the connection path between two cells that share at least one edge in common.

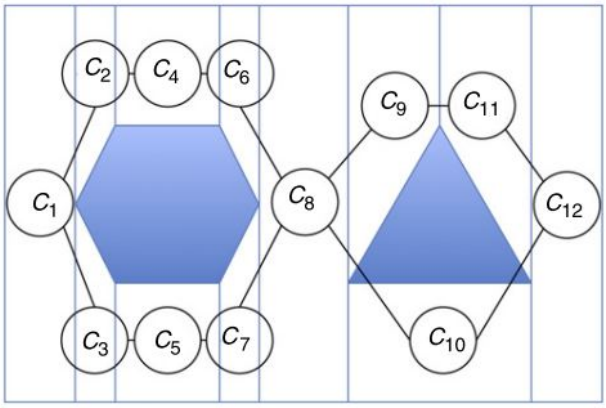


Figure 2.5: Trapezoidal decomposition of an example workspace with its corresponding adjacency graph. [7]

Therefore, a numerical index is assigned to each cell and the possible connections can be represented as segments that connect the cells (Fig. 2.5); the decomposition must be stored to trace back to all possible passages between adjacent cells.

Two important techniques are *trapezoidal decomposition* and *boustrophedon decomposition* (literally means "the way of the ox", signifying the pattern in which an ox drags a plow back and forth). The former is one of the simplest techniques [3], and involves dividing a complex area into convex cells (trapezoids), as in the case of obstacles on the scenario (Fig. 2.6a).

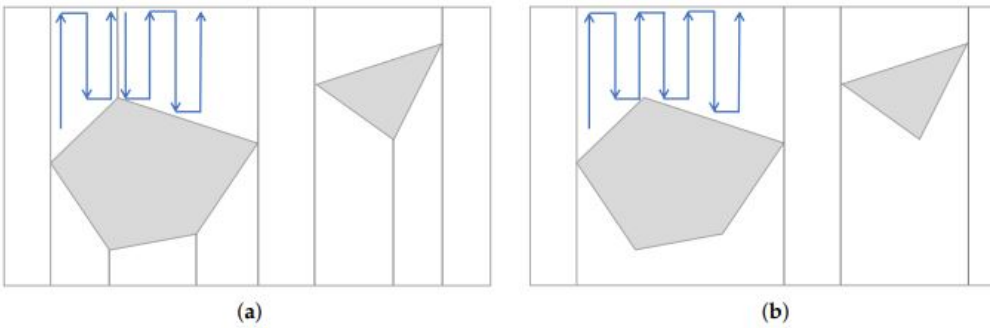


Figure 2.6: Two types of exact cellular decomposition: (a) Trapezoidal decomposition; (b) Boustrophedon decomposition. [3]

An example, by Oksanen [8], propose an off-line algorithm based on the trapezoidal decomposition for coverage path planning in the case of agricultural fields and agricultural machines. The second technique is more efficient than the first as in the decomposition with trapezoids often some cells are unnecessarily subdivided and require doubled paths. In this case, instead, more cells are merged, thus obtaining larger sub-areas and shorter paths; to do this, only the terminal extremes of the obstacles, called critical points, are considered and the cells are divided according to these (Fig. 2.6b). In both cases, it may possibly be envisaged to assign a limited number of cells to each robot of the multi-UAV system.

An interesting work on this topic is described by Torres et al. [9], in which the goal is a 3D reconstruction of a scenario using a single aerial vehicle. As shown in Fig. 2.7a-b, the entire area is divided into convex sub-areas, and for each one the optimal path is defined based on the orientation of the maneuvers (right-handed or left-handed turns Fig. 2.7c).

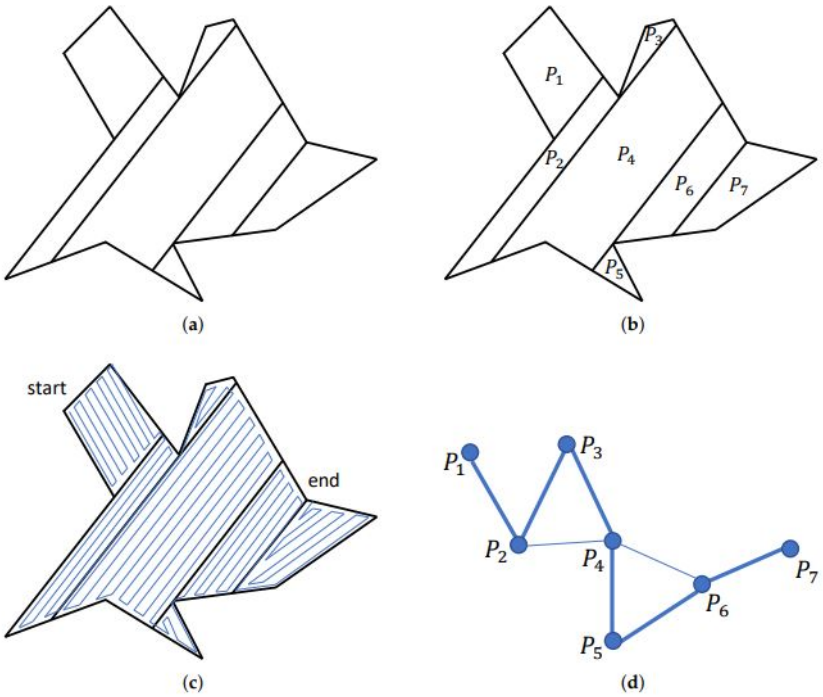


Figure 2.7: Decomposition and coverage of concave polygons: (a) Convex decomposition; (b) Sub-region combination; (c) Coverage path; (d) Undirected graph. [9]

Subsequently, the sequence of transition between adjacent cells is also optimized (Fig. 2.7d), using the connection graph, minimizing the entire path based on the starting and

ending points in the single cell.

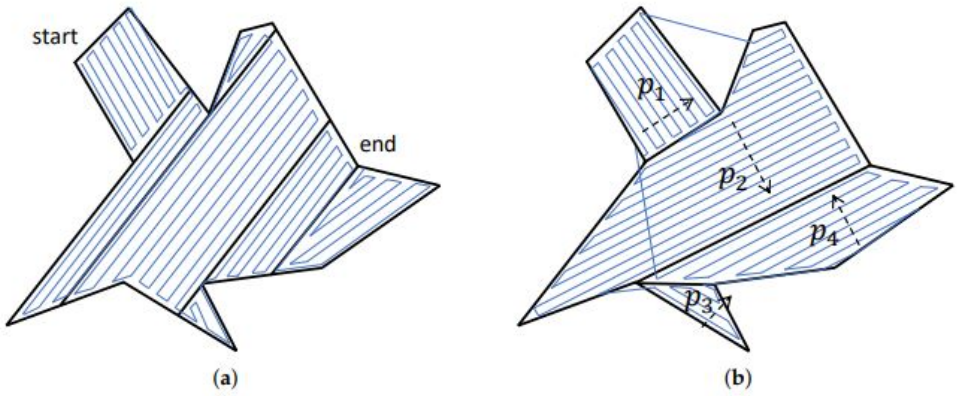


Figure 2.8: Comparison between the decomposition approaches in concave areas: (a) Convex decomposition; (b) Concave and convex decomposition. [9]

However, given the high number of cells, the computational cost linked to the possible permutations in the transitions between different areas, leads to consider a further solution (Fig. 2.8), namely the use of concave cells (as in the boustrophedon decomposition), through which, in the aforementioned work, there is a reduction in the time necessary to calculate the optimal route, and also a shorter length of the entire path.

A further technique [10], proposes the Morse-based decomposition, which overcomes the Boustrophedon decomposition with the advantage of handling also non-polygonal obstacles; By choosing different Morse functions, different cell shapes are obtained, e.g. circular or spiked cells (Fig. 2.9).

Approximate Cellular Decomposition

Is the last method, it is typically used in the case in which the UAV must necessarily pass through some waypoints for the collection of data or the scanning of a specific cell. In this case the area is divided into square cells and the path is defined by joining the centers of these cells. It is a method with numerous applications but above all in the field of RW, as the trajectories are typical of these, with movements similar to those of a pawn on the chess board, with hovering for each cell awaiting data collection.

An example in this field is presented by Barrientos et al. [11], in which the area to be

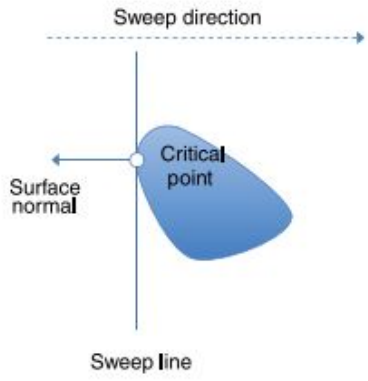


Figure 2.9: Cell boundaries in Morse decomposition are placed at critical points, where the surface normal of the obstacle is perpendicular to the sweep slice, and parallel to the sweep direction. [10]

covered (gray area in 2.10) is a portion of the rectangle discretized in cells and there is no fly zone that allows anti-collision between different robot. The algorithm allocates the area to be covered within the rectangle, then identifies the optimal number of moves to optimize the coverage of each sub-areas, as the aim is to recompose the map of the area with several photos, and the mission time of the aircraft.

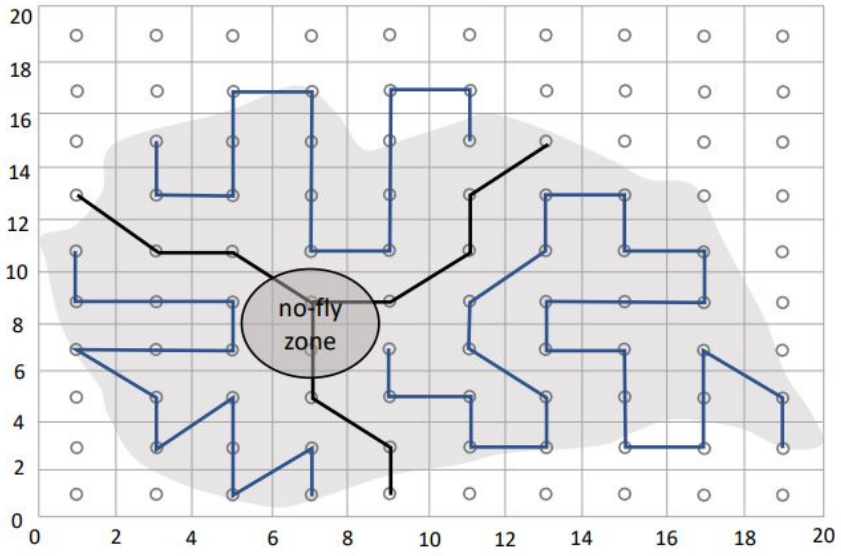


Figure 2.10: Coverage path planning in irregular-shaped areas containing no-fly zones in the subregions bounds for collision avoidance. [11]

No Decomposition	Exact Decomposition	Approximate Decomposition
Easier Algorithm	Complex Areas	Complex Areas
Basis for multiple regions	Avoid Obstacles	Avoid Obstacles
No Complex Areas	Rotor & Fixed Wing	Rotor Wing
	More Complex Algorithm	More Complex Algorithm

Table 2.1: Pros and cons of the different Decomposition approaches.

2.2.2 Geometric Based

Geometric-based methods are typically used to find the shortest Euclidean path. The most used method exploits the Voronoi diagrams, in which each point identifies the center of the Voronoi region, and the individual cells are constructed in such a way that the points inside each cell are closer to the center of the same cell than all the other possible centers (Fig. 2.11), the coverage algorithm works according with the aim of having areas proportionate to the performance of the robots (if heterogeneous), and therefore mission times that are comparable to each other. The work proposed by Kantaros [12] is a partitioning scheme that depends on creating a power diagram to cover non convex areas. The power diagram takes into account the different sensing capabilities available in each agent in addition to the visibility domain of the agents. Further information will be presented in the Chapter 3.

Figure 2.11: Voronoi diagram scheme.

2.2.3 Reward Based

Reward Based are methods used due to their advantages such as nonlinear mapping, learning ability, and parallel processing. The most important used methods include Neural Network (NN), **Nature Inspired methods** and hybrid algorithms.

Regarding bio-inspired methods, an interesting work concerns the approach presented by Ranjbar-Sahraei et al. [13] which exploits the behavior of ants. In fact, in this multi-UAV method, the individual robots release pheromones during their journey that allow to orient the paths of all the robots, because areas already covered possibly must be avoided by the others. Otherwise, areas can be scanned with different priorities depending on the presence or absence of the component in the air.

Methods of this type are increasingly used, exploiting for example also the behavior of neural networks, for which the grid of neurons is defined and the entire path is modeled on the basis of the cost function that represents how much, or less, a specific area has been visited. Also in this case, the network allows to direct the robot towards less visited areas, with the single neurons communicating between them and directing the path, with the usual aim of avoiding collisions (as an algorithm for multi-UAV) and allowing total coverage of the scenario.

2.2.4 New Best View Based

NBV approaches are usually used when no information about the model exists priori, and this approach scale better to complex real-world.

A method proposed by Manjanna S et al. [14], takes advantage of the Contour-based location selection, that is the choice of points where carry out water sampling, through the collection of images with sensors on the robots, and the identification of the contours (Fig. 2.12) of the coast in a variable prior-unknown scenario. This is a very valid method for further application of CPP in monitoring water quality (or air), and its contamination.

In this case, it is the new view that guides the path as it is not initially available, but by appropriately setting the priority to visit a certain area, it is possible to monitor the mission of the UAV.

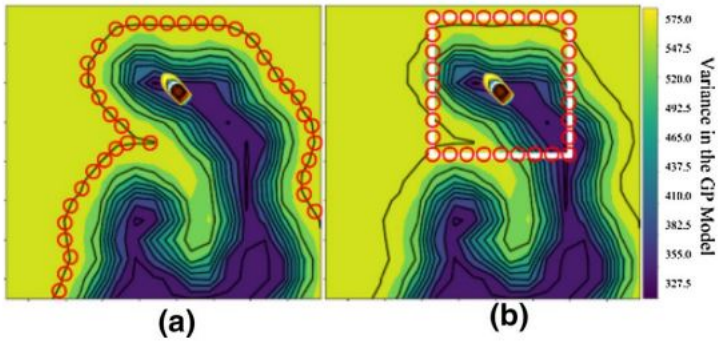


Figure 2.12: The colormap shows the variance in the spatial representation of the feld. The potential candidate locations are represented by red circles. The contours are shown in black lines. [14]

	Grid Based	Geometric Based	Reward Based	New Best View Based
Complex Areas	✓	✓	✓	✓
Basis for others models	✓	✓		
Rotor & Fixed Wing	✓	✓	✓	✓
Minimize Mission Time			✓	✓
Real Time Simulation	✓	✓	✓	✓
Unknown Scenario			✓	✓
Non-Linear Mapping			✓	✓
Learning Ability			✓	

Table 2.2: Characteristics of the different Path Planning approaches.

2.3 Multi-UAV Path Planning

The design of multi-UAV coverage is necessary when the complexity of the area to be observed increases, or the region is too large for the autonomy and velocity of the drone. In this case, typically, one or more sub-areas are assigned to individual drones, with the aim of covering the entire scenario in the shortest time.

The main challenges of this topic are:

- **Cooperation:** the algorithm is not the simple sum of several drones covering different areas, but cooperation between them is necessary to achieve a single goal;

- **Heterogeneity:** it is possible that the drones may be different from each other, with different speed performance, maximum flight altitude, autonomy. Furthermore, the sensors (for example the field of view for image capture) may also differ, or land (UGV) and sea (AUV) drones may be included. It is therefore necessary to optimize the path planning according to the different characteristic;

- **Prioritization:** for real missions, it is often useful to give priority to the coverage of certain areas rather than others, for example in case of disaster areas, so different weights will be assigned when optimizing the path;

- **Robustness:** it is necessary to include failure messages in the algorithm for lack of communication, or reconfigure the planning if there is a failure of a robot in order to cover the area that the latter should have sifted;

- **Communication:** it is necessary to consider the type of communication (centralized, distributed, etc.) and the effective range of availability of the individual drones for their communication;

- **Adaptability:** the algorithm must allow to use different scenarios, with dynamic environment, and to adapt to different situations in wide covering areas.

The use of multiple subjects introduces the problem of avoiding collisions, and, depending on the mission, also avoiding that two or more drones needlessly retrace the same area, through communication and allocation of information. Some peculiarities are:

- **Collision:** is usually avoided by considering the flight of several UAVs at different altitudes, however it is not always possible, and it is not congenial to the performance of drones on the market. More advanced method is "Sense and Avoid" [15] based on the use of video processing techniques and other sensors onboard the UAV, but in this thesis they will not be studied given the additional computational cost to consider.

- **Communication and task allocation:** the common assumption in literature is that communication is unlimited in range and bandwidth. Usually many algorithms assume that all vehicles can communicate with the central controller in order to have global

communication. However, in disaster scenarios, communication is one of the most relevant problems as it is very disturbed, therefore decentralized communications [16] have developed, between robots in line of sight (LOS) that communicate the position via their coordinates. Sometimes service drones are used in parallel with research drones, with the only task of maintaining efficient communication and the allocation of information received from coverage robots. Furthermore, using decentralized systems reduce the network overhead introduced by information exchange between robots.

One of the works available in the literature, by Maza and Ollero [17] exploits the back-and-forth motion for a fleet of heterogeneous drones. A ground station decomposes the entire area and assigns each UAV a section based on its initial positions and its capabilities (Fig. 2.13). The entire path planning is aimed at minimizing the number of maneuvers, which, as already mentioned, is an indication of lower energy consumption. Furthermore, it is possible to reconfigure the areas when one or more drones fail, giving robustness to the algorithm.

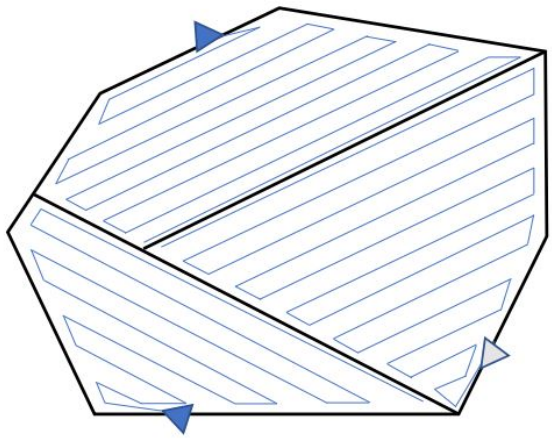


Figure 2.13: Cooperative coverage in convex polygon area using a team of heterogeneous UAVs. [17]

A different approach is instead the one proposed in the work of Vincent and Rubin [18], in which the drones travel parallel, with rectilinear trajectories. In this case, a simplified communication is considered, given the proximity between the drones (Fig. 2.14), furthermore a failure in communication is translated directly into a failure of the drone, with consequent reconfiguration. The mission is based on some criteria, such as maximize the probability of

detecting a target, give robustness to the process, minimize the number of UAVs needed and the number of maneuvers performed.

Figure 2.14: Cooperative coverage strategy in rectangular areas of interest with intelligent targets. [18]

2.3.1 Multi-UAV Spanning Tree Coverage

The Spanning Tree Coverage (STC) method was generalized to multi-robot teams by Hazon and Kaminka [19]. The work by Fazli et al. [20] proposes a spanning tree approach plans to split the scenario into a forest consisting of partial trees (Fig. 2.15). The graph is generated by performing subsequent Trapezoidal decomposition (convex polygons) of the 2D environment until each single guard (sample) can cover a corresponding single convex polygon, that is until the branch reaches the end of the area to be covered. Then, a Constrained Spanning Tour (CST) method is used to build cycle on each partial tree and assign each cycle to a robot.

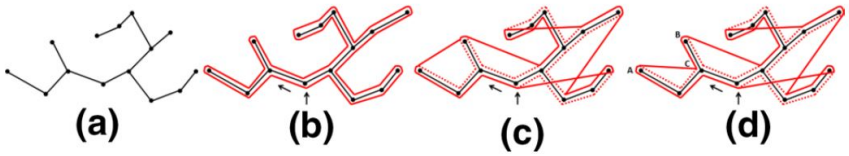


Figure 2.15: (a) A sample tree; (b) double-minimum spanning tree (DMST); (c) revised-DMST; (d) CST. [20]

In another more recent work by Kapoutsis AC et al. [21], the algorithm proposes to divide the area of interest based on the initial positions of the robots. For each robot planning, the

algorithm construct a Minimum Spanning Tree (MST) for all the unblocked nodes and then apply the ST to the original terrain and circumnavigate the robot. However, in some cases this kind of path planning encounters problems with cells with more complex geometry.

2.3.2 Multi-UAV Bio-Inspired Path Planning

One of the approaches that is increasingly used in the field of control algorithms for coverage with multi-drone systems is that inspired by nature. In many cases, with this type of development, it is possible to comply with scenarios in which it is necessary to use a control in real time, moreover, especially for the "colonies" of drones, very often nature can give us inspiration. Another advantage is that of being able to introduce the possibility of monitoring any intruders along the route (security or military missions) using specific techniques. Obviously, on the other hand, there is a computational disadvantage in many studies, however once this difficulty is overcome the overall performance is better.

The two main bio-inspired approaches are via neural networks or animal colonies, particularly those of ants.

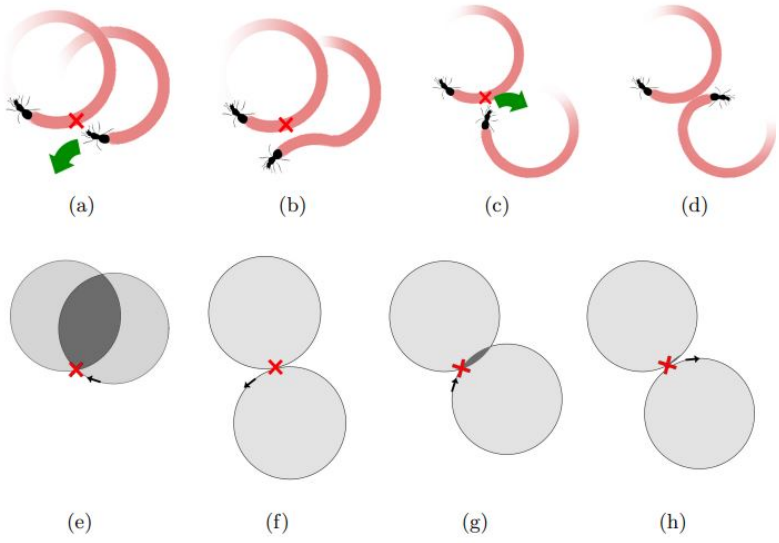


Figure 2.16: StiCo coordination principle: (a)-(b) before and after pheromone detection by internal sensor; (c)-(d) before and after pheromone detection by external sensor; (e)-(f) covered area before and after pheromone detection by internal sensor; (g)-(h) covered area before and after pheromone detection by external sensor. [13]

Neural Network-Based Approach

The approach based on neural networks, allows to obtain the mission path avoiding collisions, and maximizing total coverage. It also allows to optimize the search for a specific point (surveillance mission), using the fastest route available. As in the work of Godio et al. [22], the network is made up of a series of neurons, connected to the adjacent ones (Fig. 2.17), which, through a cost function, allow to increase the visitability of neurons not yet visited. Therefore, translating into coverage terms, the overall development is to direct the flight through the sub-areas (neurons) not yet visited. Furthermore, this approach is often learning based, and the robot can effectively avoid obstacles and walls in unknown environment, but, due to the computational complexity for the learning, sometimes has difficulty dealing with unstructured environments.

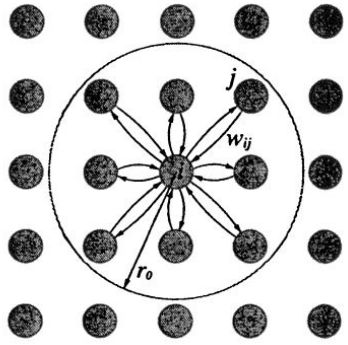


Figure 2.17: Schematic diagram of a neural network for complete coverage path planning.

Ant Colony approach

The Ant Colony Model is inspired by the foraging behaviour of ants that they can obtain the shortest path between their nest and food. In the reference case study [23], there is a group of heterogeneous UAVs, and the development consists of three main steps: subdivision of the scenario into smaller regions, assignment of regions to individual drones, optimization of the order of visit of these. Furthermore, the collision is avoided through the release of a pheromone by the drone, which, if heard by others, avoids covering the same region. Again with an ant-inspired approach is the work of Ranjbar-Sahraei [13], in which, however, the motion of the robots (ants) is constantly circular, because they are UGVs (Fig. 2.16). In

this case, the extension of the model in the case of intruders (ID-StiCO), which orient the path of the drones in search of these, is very interesting, with typical mission purposes of surveillance.

For the choice of the approach model, it was decided to opt for the system inspired by ant colonies which, from a data collection of previous control algorithms, seems to be better in terms of computational cost than models based on neural networks and spanning tree. As regards the coverage percentages, however, the methods are comparable on the whole. The interesting work cited also focuses on the performance of quite heterogeneous UAVs, as well as on the optimization of the visit order, orienting the final choice. Finally, the approach with a network of nodes/neurons is often used for UAVs with rotary wings rather than fixed wings as in the case of this thesis, limiting the possible maneuvers.

In Chapter 4 the model will be further analyzed, with the proposal of a hybrid solution in the different sections of the project, thus departing from the work previously presented, exploiting for the subdivision of the area with the boustrophedon decomposition initially, and subsequently the subdivision exploiting the Voronoi diagram (Chapter 3) for grid partition, and the ant colony system method for regions' allocation and order of visiting.

	Grid partition	**Allocation of Regions**	**Order of visiting**
Method	- Boustrophedon - Delaunay Triangulation - Voronoi diagram	Ant Colony	Ant Colony

Table 2.3: Summary of selected methods.

Chapter 3

Area Decomposition

For the subdivision of the areas of the region to be covered, two different solutions were evaluated during the project, opting for the second as definitive as it is more efficient for a system of several drones. The **Boustrophedon decomposition** is fundamentally based on the presence or absence of obstacles in the map, and is fallacious when the obstacles are few in number or much smaller than the characteristic lengths of the total area (as demonstrate the results in the Section 5.3), instead the subdivision with the **Voronoi diagram** considers a number of points a priori (chosen randomly or, as described below, based on Delaunay triangulation).

In any case, the coverage algorithm has as input an initial map, for simplicity described by a rectangle (a rectangular image of size $M \times N$), with obstacles inside. Typically in this type of applications the obstacles are black polygons on a white background (as in the Figure 3.1) which allow, through the conversion to a binary occupancy grid image, to associate the presence or absence of an obstacle in a pixel with the corresponding binary number (0 for white and 1 for black obstacles). Therefore, all the following work starts from the assumption of having a matrix of pixels $M \times N$ in which it is possible to fly or not according to the binary value. The type of obstacles can be very varied as in the examples presented in the figures, however in the second phase simple convex and non-concave obstacles will be assumed, although concave obstacles can also be described by convex obstacles concatenated.

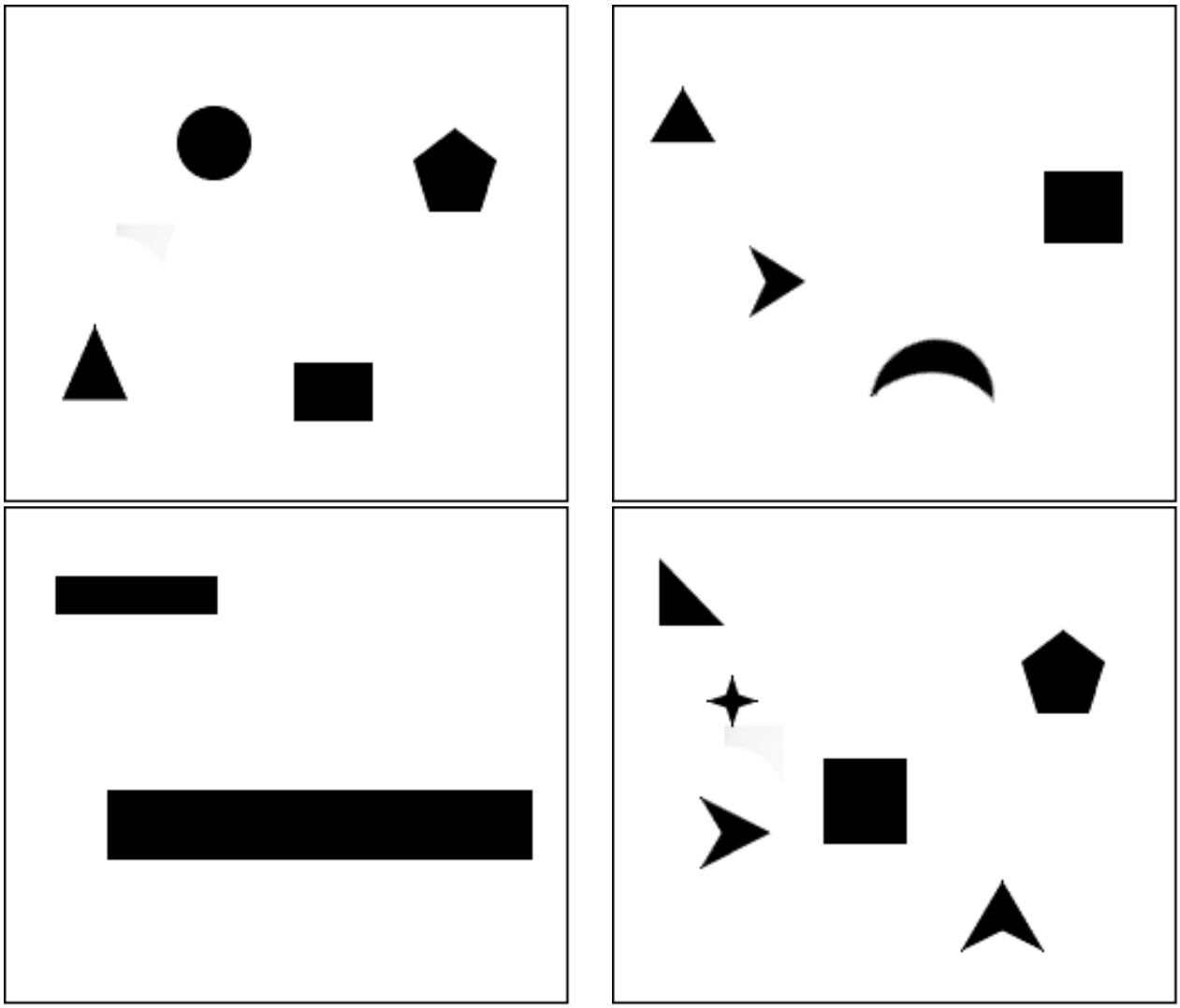


Figure 3.1: Example of image for Map.

3.1 Boustrophedon Decomposition

The Boustrophedon subdivision overcomes the primordial trapezoidal subdivision by providing for the creation of distinct cells whenever a critical point is encountered. As can be seen in Fig. 3.2, passing from left to right, every time an obstacle is encountered, the cells are split up to the end of the obstacle itself.

In this work, it was implemented with the following steps (as in [9]):

- Reading the image one slice at a time (a column one pixel wide), and observing if the pixels always had a binary value equal to 0, therefore no obstacles, or switched to the

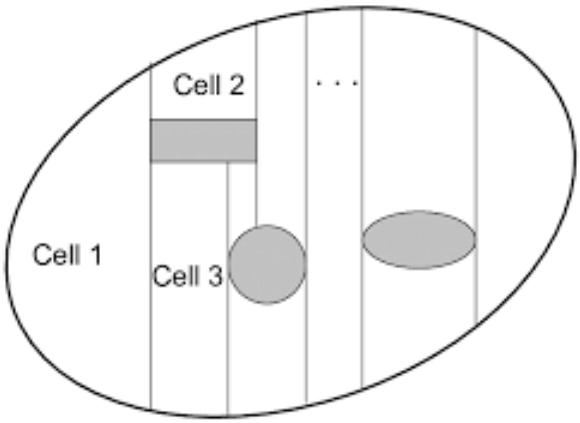


Figure 3.2: Scheme of Boustrophedon Decomposition

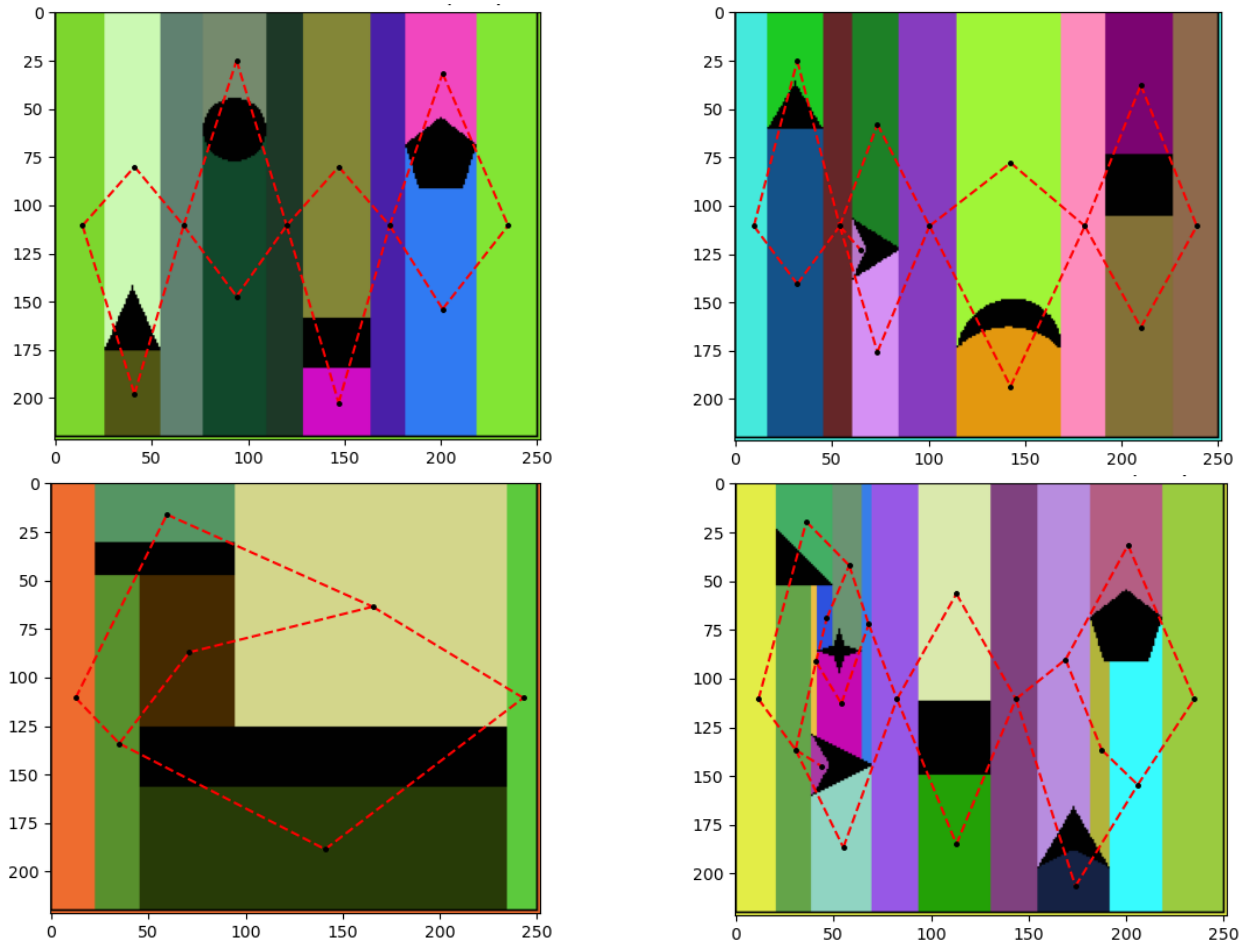


Figure 3.3: Example of Boustrophedon decomposition. Black dots are the centroids of cells (with random different colors areas), red dotted lines for adjacent cells.

value equal to 1, thus registering the presence of an obstacle;

- Going forward in scanning the image, if the obstacle is extended over several columns, the entire column is divided into two cells, just as in the examples described in the previous Chapter, or in the case of several obstacles, each region is further divided;
- At the end of the obstacle, the two previously divided cells come together and the decomposition is continued until the end of the image.

At the end it is possible to visualize the whole image with the subdivision into cells, the presence of the centroids for each cell (black dots) and the adjacency graph, that is a series of lines (red dotted) that connect the different cells adjacent to each other (adjacent cells has coincident boundaries and therefore it is possible to move between one and the other).

From the following images (Fig. 3.3 corresponding to the previous maps), it is possible to observe how this type of subdivision works quite well with concave obstacles, but nevertheless, in the case of a few obstacles, or of small size ones compared to the entire area, it creates cells that are too elongated or thread-like, difficult to be covered by a single drone.

3.2 Voronoi Diagram Decomposition

The applications of the Voronoi diagram are many due to its applicability to the tessellation of areas; in the meteorological and hydrological field it is used for the collection of precipitation data in a specific area, in the biological field for the structures of cells and bones, in ecology for the growth patterns of forests and fires, or to define specific meshes, for specific microstructures of metal alloys, for urban planning etc.

In mathematics, the Voronoi diagram is a subdivision of space into several cells given a set of points (called seeds, sites, or generators); it is used in several scientific and technological fields. For each seeds the cell is constructed so that all points of this are closer to the seed of the cell than any other.

Formally, let X be a metric space with distance function d, let K be a set of indices and let $(P_k)_{k\in K}$ be a tuple (ordered collection) of nonempty subsets (the sites) in the space X.

Then, the Voronoi region is defined by:

$$R_k = \{x \in X \mid d(x, P_k) \leq d(x, P_j) \quad \forall j \neq k\}$$

In this thesis, the choice of the points for the generation of the cells was initially based on a random choice between all the points available and flyable in the area, subsequently, observing, especially around the narrowest obstacles, that cells wrapped the obstacles, a different solution was chosen, based on the Triangulation of Delaunay. Some examples of tessellation based on random generation are depicted in Figure 3.4a, where the number of cells was proportional to active UAVs and number of detected obstacles, and in Figure 3.4b where there are the circled red cells that oriented the next choice as they are very difficult to cover by a drone if not "jumping" on the other side of the obstacle.

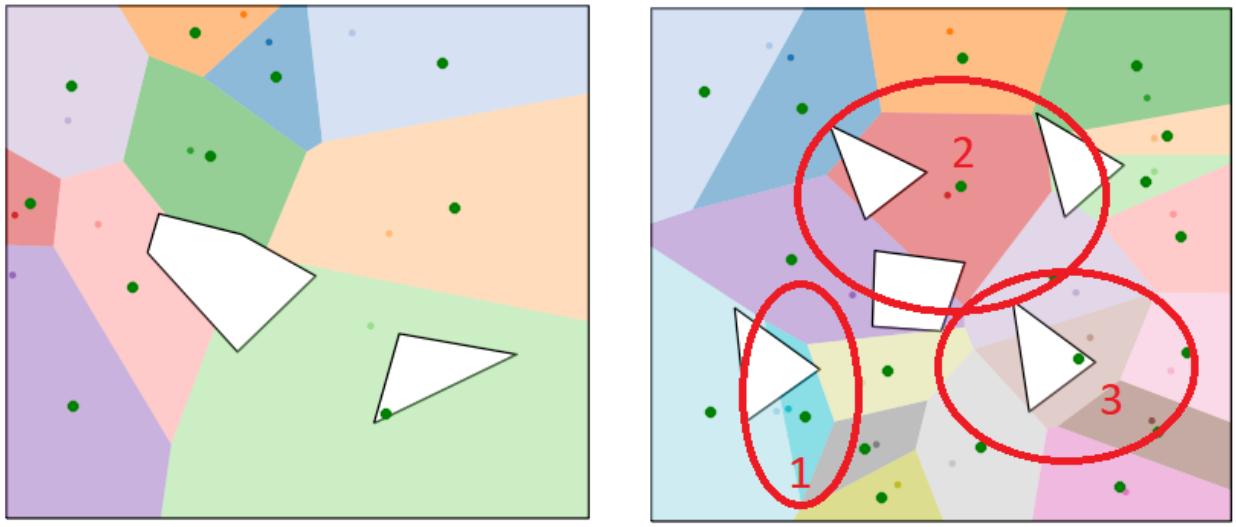


Figure 3.4: Example of Voronoi diagram with random choice. Little dots are the point of the set P_k, big green dots are centroids of cells

3.2.1 Delaunay Triangulation

Delaunay triangulation is especially used in computational geometry for mesh generation. Starting from a set of points G_k, triangles are generated so that all points are outside the circumcircle of any final triangle, in order to maximize the minimum of all angles of the triangles (Fig. 3.5). Furthermore, the triangulation is the dual graph of the Voronoi diagram,

as the circumcenters of the Delaunay triangles correspond to the vertices of the cells generated with the Voronoi diagram if the same set of points is used, or rather the vertices of the Delaunay triangles correspond with the set of points P_k mentioned above for the Voronoi diagram (Fig. 3.6). This particular dualism therefore allowed to subdivide the entire area initially with a Delaunay triangulation, subsequently the same vertices were exploited as input to the previous function of the Voronoi diagram, avoiding the random generation of the cells.

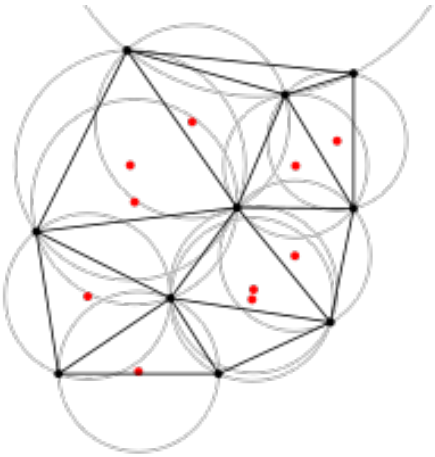


Figure 3.5: The Delaunay triangulation with all the circumcircles and their centers (in red).

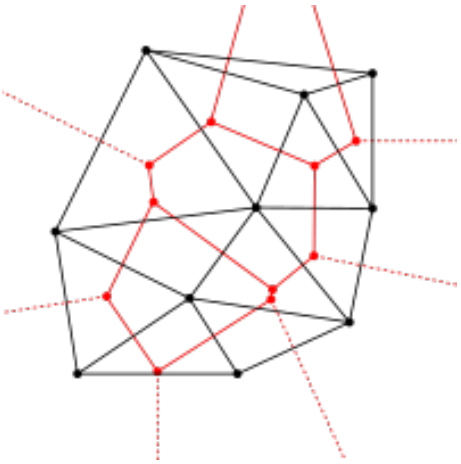


Figure 3.6: Connecting the centers of the circumcircles produces the Voronoi diagram (in red).

In particular, the choice for triangulation was carried out with successive steps, with the same tool as explained in the Section 5.1:

- **Pure triangulation:** only for a set of points, without holes, is useless for this application;
- **Constrained Delaunay triangulation CDT:** only with the vertices of the image and of the obstacles, creates stranger triangles and it is the worst (Fig. 3.7);
- **Conforming constrained Delaunay triangulation CCDT:** uses the set of vertices and add the Steiner points, that are additional points chosen for a better solution (Fig. 3.8);

- **Conforming constrained Delaunay triangulation maintaining the Delaunay triangles:** is identical to the CCDT but the triangles are all Delaunay triangles, condition lost with Steiner points (Fig. 3.9);
- **Conforming constrained triangulation maintaining the Delaunay triangles and with a constraint on the maximum area:** as the previous but the maximum area of triangles is imposed by the algorithm/user (Fig. 3.10).

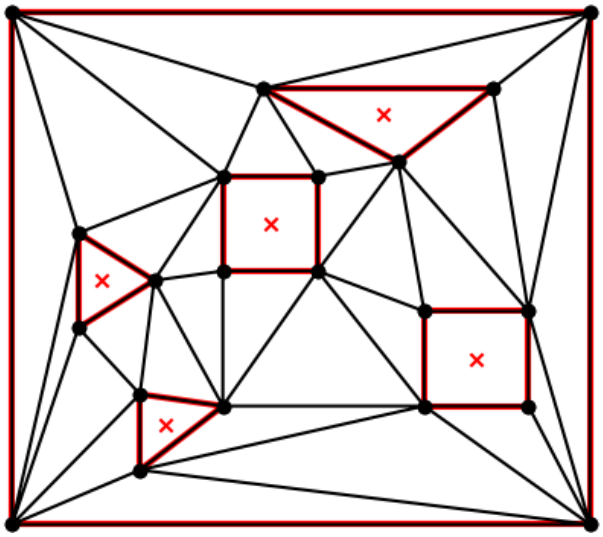


Figure 3.7: Constrained triangulation CDT.

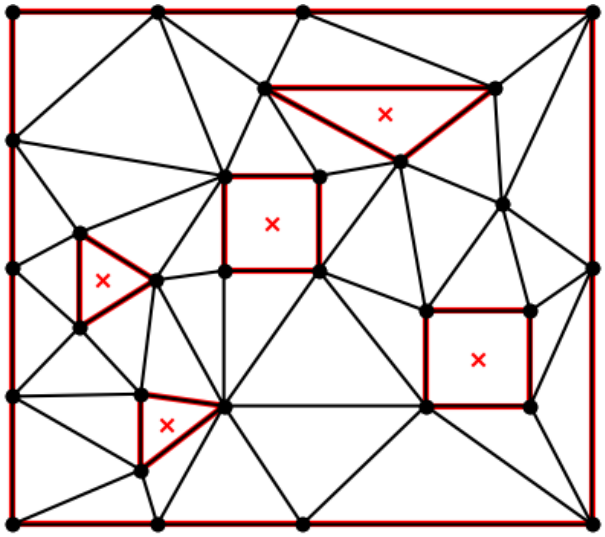


Figure 3.8: Conforming constrained triangulation.

The subdivision through the use of the Delaunay triangulation allows, unlike the Boustrophedon decomposition, to subdivide even areas in the absence of obstacles, an important feature in the case of a multiple drone system, because, even in the absence of no fly zone, it is necessary divide the entire rectangle to be covered (Fig. 3.11)

The final configuration as in Figure 3.13, after some tests, was that for which the constraint of the parameter of the area of the triangles was proportionally linked to the obstacle of greater area, because, on one hand, choosing the smallest obstacle involves cells that are too small in areas where it is not necessary to thicken the areas (as in Figure 3.12), on the other hand, choosing the largest obstacle does not affect the minors because in that case the condensation is dictated by the presence of the vertices of the obstacles, moreover, this constraint is necessary to avoid complex cells in presence of very squashed triangular

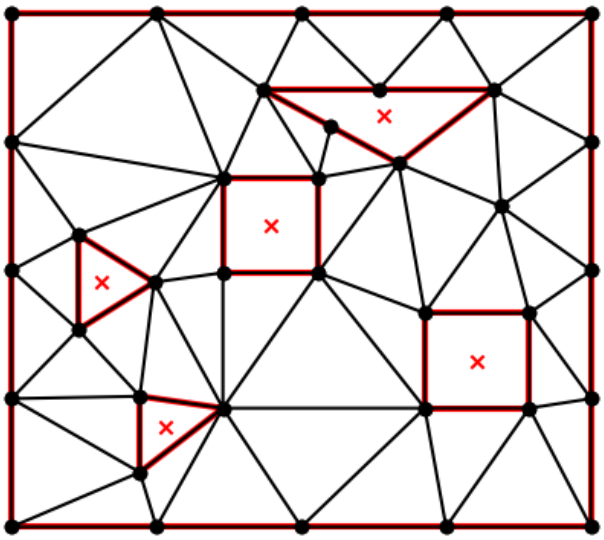


Figure 3.9: Conforming constrained Delaunay triangulation maintaining the Delaunay triangles.

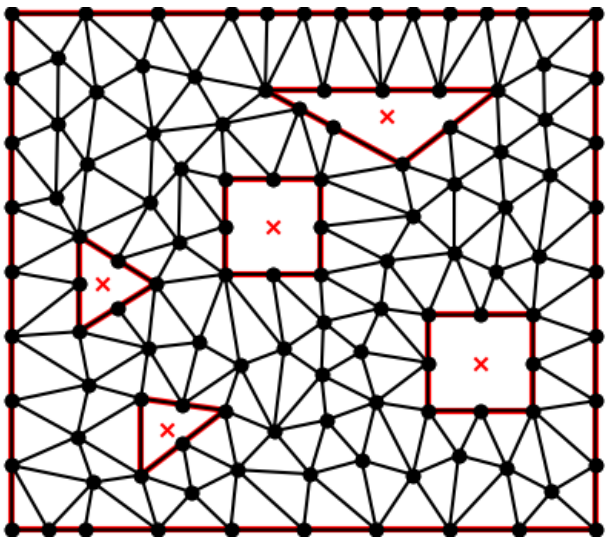
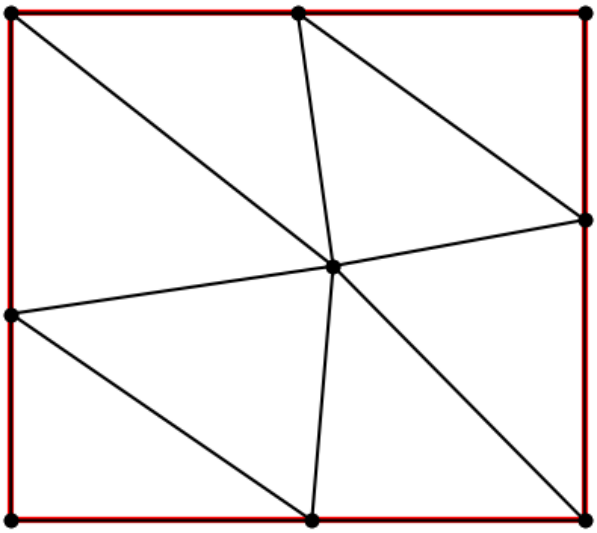


Figure 3.10: Conforming constrained triangulation maintaining the Delaunay triangles and with a constraint on the maximum area.



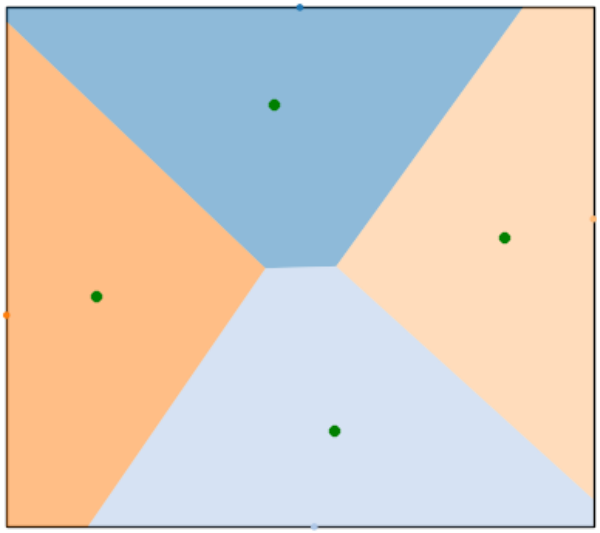


Figure 3.11: Conforming constrained Delaunay triangulation with a constraint on the maximum area and the dual Voronoi diagram for 0 obstacles.

obstacles. Further investigation on the constraint on the maximum area will be discussed in the Section 5.3.

	Boustrophedon	Voronoi Random	Delaunay + Voronoi
Pro	- Good with non-convex obstacles - Easier implementation	- Dense cells - Link with n° UAV/Obstacles	- Dense cells - Link with n° UAV/Obstacles - Link with dimension and vertex of Obstacles
Cons	- Coarsed cells - No link with n° UAV	- Presence of weak cells - Random generation of cells	- Harder implementation

Table 3.1: Pro and cons of different Decomposition on Python

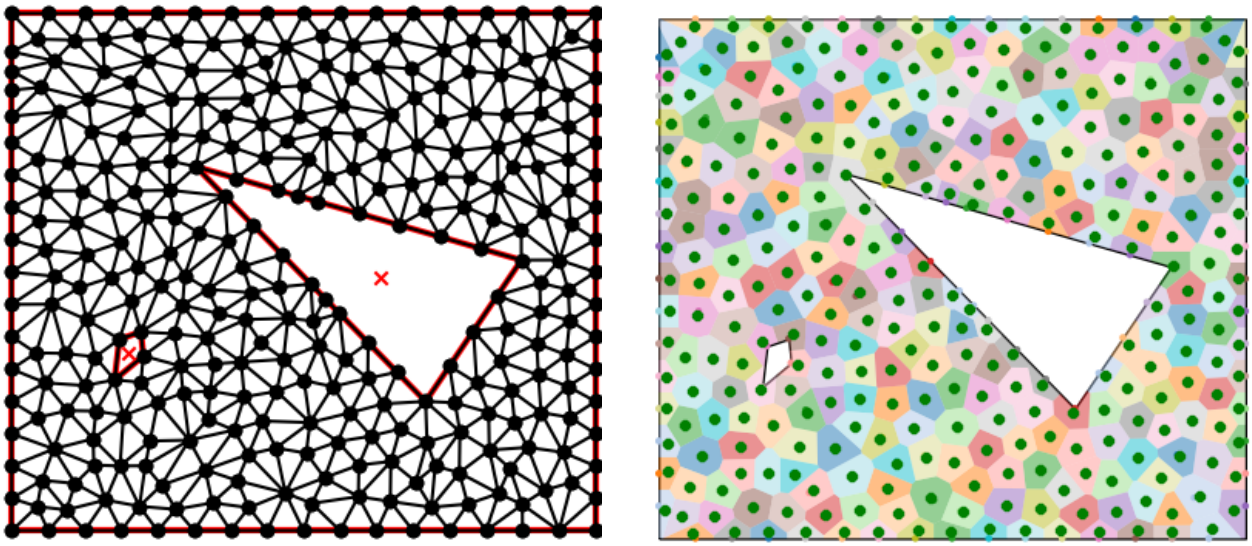


Figure 3.12: Wrong conforming constrained Delaunay triangulation with a constraint on the maximum area and the dual Voronoi diagram.

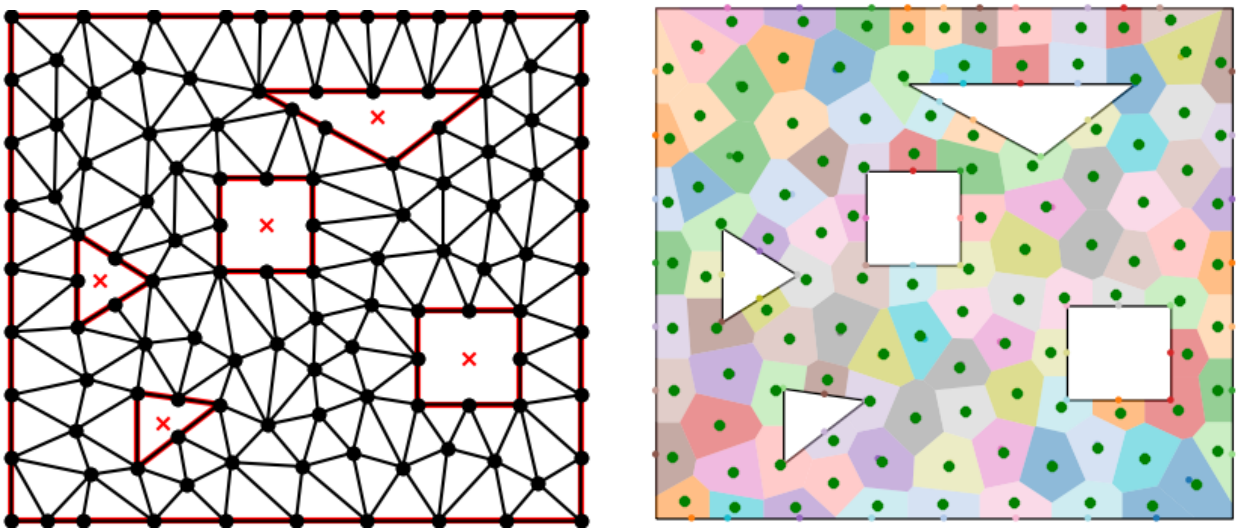


Figure 3.13: Final conforming constrained Delaunay triangulation with a constraint on the maximum area and the dual Voronoi diagram.

Chapter 4

Ant Colony System

The Ant Colony System (ACS) is an optimization algorithm applied to the traveling salesman problem (TSP), proposed for the first time in the work done by Dorigo M. and Gambardella L. [24], which exploits the natural behavior of ants in evaluating the optimal visit path for more cities. In the specific application, the algorithm is used for the optimization of the visit order of the different cells coming from the decomposition of the entire region, with the allocation of the single sub-regions to the different UAVs according to the *Effective Time Ratio*, as proposed in the work of Chen et al. [23].

4.1 Natural Behaviour

The natural metaphor used in the algorithm is that of the ants in their colonies which, through the exploitation of the pheromone released by them in their path, are able to reach the shortest path to their nest without using the view, drawing paths in perfect line even after some time. The more pheromone is deposited on the ground, the more the path has been used by the companions of the colony, reaching the shortest possible path. In Fig. 4.1, the whole mechanism is observed, in particular in Figure *1a*, the ant L1 and R1 find themselves at a crossroads, represented by the choice between two cells for an UAV, and in the first instance they have to decide without further information which path to take, therefore randomly. Half of the ants traveling left to right (L names) and half of the ants traveling right to left (R names) are expected to choose the paths above, the remainder the opposite path. Then,

according to Figure *1b-c*, the number of subsequent paths is proportional to the amount of pheromone released; since the lower path is shorter than the upper one, more pheromone will remain in the air before it evaporates and more ants will visit it on average, and therefore pheromone accumulates faster *1d*. As time progresses, ants will choose the shortest path more and more easily, until they reach a probability of 100%. By transposing this behavior to the drone case study, the information of the pheromone is exploited by "depositing it" on the boundaries of the cells, the communication bridges between them, using a set of ants that visit the different cells in a precise order, performing the parallel search for good solutions to the TSP, exploiting the pheromone information and the heuristic information, based on the length of the travel path between one cell and the next, until convergence and optimal order of visit are obtained.

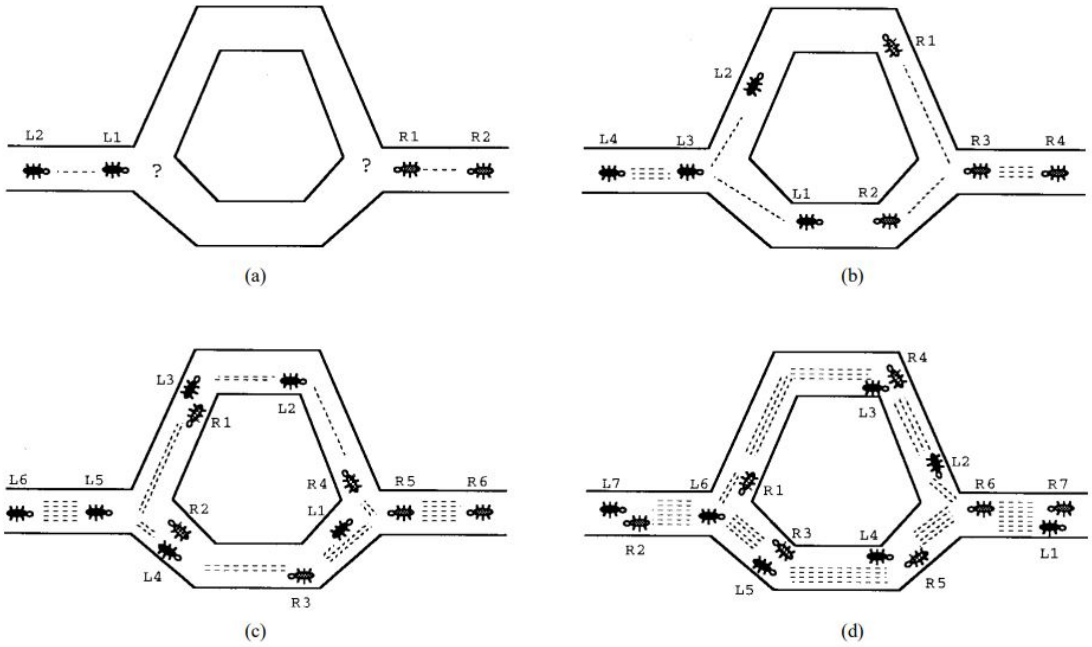


Figure 4.1: Schematic natural behaviour of ant colony. [24]

4.2 Ant Colony Algorithm

For the specific algorithm, the focus is on the optimization of the allocation of the regions in the set of heterogeneous UAVs and on the optimization of the visit order of the same, however

the pheromone information could also be used to avoid collisions, as a high concentration of this should reorient the path avoiding clashes between the various drones. In this case, the collision is mainly avoided through one of the allocation constraints, that is the visit of a cell only by a single UAV, without repetitions or identical scanning paths between multiple drones, however traffic avoidance could be included for as regards the parallel flight between different cells, with paths that could also intersect.

4.2.1 System Models

The system is composed of a set of UAVs, $U = \{U_1, U_2, ..., U_n\}$, with user-imposed characteristics, and a set of regions to visit $R = \{R_1, R_2, ..., R_m\}$, coming from the decomposition into cells of the entire area to be covered. Each UAV is characterized by 3 fundamental characteristics, speed, altitude and field of view (FOV) of the sensor used $U_i = \{V, H, FOV\}$, but in reality the altitude is not used for optimization purposes, even if it can link the speed and FOV of the drone. Furthermore, for each robot there is the information of the initial position, according to the two Cartesian coordinates on the ground $\{X_i, Y_i\}$, information used to optimize the first cell visited and the further neighboring cells. As for the regions, a $D_{j,k}$ matrix is constructed, which represents the distances between the region R_j and R_k, therefore the matrix has a zero diagonal (distance between cell R_j and $R_{k=j}$ is equal to zero), and is symmetrical, since the path between R_j and R_k is the same between R_k and R_j, calculating the distances between the cartesian coordinates of the centroids of each region.

Parameter	**Meaning**
U_i	Generic UAV
R_m	Generic Region to be visited
V, H, FOV	Velocity, Altitude and Field of View of each UAV
$P(x_i, y_i)$	Initial Position of UAV
$D_{j,k}$	Matrix of Distances between regions

Table 4.1: Summary of selected parameters.

Using the speed and FOV parameters, it is possible to calculate the **scanning time**

and the **flight time**, subsequently used for the optimization of the allocation through the *Effective Time Ratio*. The former is the time required to cover a single region by each UAV, and the latter is the time required to transfer from one region to another. For the scanning time a $TS_{i,j}$ matrix is composed, and for each UAV it is calculated as the area of the region (m^2) divided by speed (m/s) and FOV (m). Instead, for the flight time, a $TF_{i,j,k}$ matrix is obtained, calculated as simply the distance between the centroids of the reference regions (m), divided by the flight speed (m/s).

$$\forall\ i,j\ \Rightarrow\ TS_{i,j} = \frac{A_j}{V_i \cdot FOV_i} \qquad\qquad \forall\ i,j,k\ \Rightarrow\ TF_{i,j,k} = \frac{D_{j,k}}{V_i}$$

Constraints

For the chosen algorithm, there are some constraints during optimization:

- The UAV has a set of regions assigned to it and must visit all of them without ever returning to the initial flight region before they are finished;
- The number of UAVs adopted to perform the coverage task is less than or equal to the total number of UAVs;
- Each UAV departs from the region in which it is initially located, and if it is in a no-fly-zone it cannot be chosen for region allocation;
- Every region should be scanned by UAVs once and only once, in order to ensure that no region is scanned repeatedly.

4.2.2 ACS Planning Algorithm

The algorithm proposed for the planning of the coverage is divided into two main phases, the first of **Region Allocation**, taking up the work proposed by Chen et al. [23] for a multi-UAV system, the second of **Order Optimization**, based on the aforementioned Dorido and Gambardella work, also taken up by Chen.

Region Allocation

In this phase, starting from the regions to be visited, obtained directly from cellular decomposition, the sets of regions for each UAV are obtained, exploiting the specific characteristics and the previously calculated time to scan and time to fly values. In particular, the allocation is optimized considering the *Effective Time Ratio*, that is an index that evaluates the efficiency of scanning with respect to flying. It, for each pair of regions R_j and R_k, initial and final region of a transition, is obtained as the ratio between the scanning time of the arrival region $TS_{j,k}$, and the sum of this time with the flight time between the departure region and the arrival one $TF_{i,j,k}$, for each UAV available to fly. The rate therefore increases not only with the scanning area, but also decreases for closer regions, and therefore close to the initial position of the drone.

Hence the allocation is made by maximizing this parameter, region with larger scan areas and closer to free UAV are more likely selected, typically composing sets of regions that surround the initial position of the drone, mainly depending on the product of velocity and field of view of the specific U_i.

$$\forall i, j, k \quad \Rightarrow \quad ETR_{i,j,k} = \frac{TS_{j,k}}{TS_{j,k} + TF_{i,j,k}} = \frac{A_k \cdot V_i}{V_i \cdot FOV_i \cdot D_{j,k} + A_k \cdot V_i}$$

Order Optimization

In the previous phase only the set of regions to visit for each UAV was defined, however it is necessary to optimize the order of visit of the same in order to have the shortest possible path. It is in this phase that the potential of the Ant Colony system algorithm is exploited, through the use of two information, the heuristic one $\eta(j, k)$ for each pair of regions and the one based on the pheromone $\tau(j, k)$, which is updated by artificial ants and represents a degree of desirability of the edge of the regions. In the case of an anti symmetric traveling salesman problem $\eta(j, k) \neq \eta(k, j)$, the transition from a region j to a k is not identical to that from k to j, this occurs in the presence of several regions or for obligatory direction to cover the set of these.

The ACS works as follows: each ant individually generates a tour of the regions according

Algorithm 1 Pseudo-code of Region Allocation [23]

Input: Uav set U and Region set R

Output: Region for each UAV

$LeftRegion \leftarrow$ R;
$\forall i \in [1, n]$, $Region[i] \leftarrow 0$;
while $LeftRegion \neq 0$ **do**
    $i \leftarrow$ index of the UAV which would complete the coverage task at the earliest time;
    $j \leftarrow$ index of the last region that would be visited by U_i;
    Calculate effective time ratios of regions in $LeftRegion$;
    $k \leftarrow$ index of the region that has the largest effective time ratio;
    Remove R_k from $LeftRegion$;
    Add R_k into $Region[i]$;
end while

to a **State Transition Rule**, preferring the closest regions and with a high pheromone content, just like the natural behavior mentioned. During this phase the pheromone content is updated whenever there is a transition of regions (*local updating rule*), and when all the ants complete the tour there is a further update of the pheromone for the shortest tour (*global updating rule*). So both heuristic and pheromone information drive subsequent tours of region order optimization.

1) State Transition Rule: every time an l ant has to choose the next region R_k to visit, it is subject to the rule 4.1.

$$R_k = \begin{cases} argmax_{R_k \in \Pi_a}\{[\tau(j,k)]^{\alpha} \cdot [\eta(j,k)]^{\beta}\}, & if \quad q \leq q_0 \\ S, & otherwise \end{cases} \tag{4.1}$$

At each step, a value of q between 0 and 1 is randomly generated, if this is lower than the q_0 threshold, the pheromone information and heuristics are exploited ("exploitation"), if instead the value is higher, the "exploration" is random according to a precise probability distribution $p_l(j,k)$. The parameter q_0 therefore determines the degree of importance of exploration with respect to exploitation, the smaller it is, the less the information collected

by the colony is exploited. The $\tau(j,k)$ parameter represents the pheromone information, and is an $m \times m$ matrix as it associates the boundaries between each couple of region, while the $\eta(j,k)$ parameter is related to the inverse of the distances between the two regions. The α and β exponents regulate the relative importance between the heuristic information and that of the pheromone, typically the values are $\alpha = 1$ and $\beta = 2$ [24], but further tests will be presented in the next Section 5.3. While Π_a is the set of available regions to be visited for each ant.

Through the transition rule the the pheromone on edge is multiplied by the corresponding heuristic value to favor the choice of edges which are shorter and which have a greater amount of pheromone. Globally, the transition rule is pseudo-randomic-proportional, as it exploits in both cases (4.1 - 4.2 above or below threshold) the information collected during the iterations.

2) Random Wheel Rule: for the probability distribution to visit R_k starting from R_j, in the case in which q is higher than the threshold, for each region R_k available to be visited ($k \in \Pi_a$), a weight for the Russian Wheel is obtained according to formula 4.2, always exploiting the information collected previously. This choice helps also to avoid the premature convergence and to explore different tour.

$$p_l(j,k) = \begin{cases} \frac{[\tau(j,k)]^\alpha \cdot [\eta(j,k)]^\beta}{\sum_{R_k \in j_l} [\tau(j,k)]^\alpha \cdot [\eta(j,k)]^\beta}, & if \quad R_k \in \Pi_a \\ 0, & otherwise \end{cases} \tag{4.2}$$

3) Initialization: as for the work of Dorigo and Gambardella [24], the initial values of the pheromone are identical and equal to τ_0, as in equation 4.3, where L is the optimal tour obtained from the approach of the closest neighbor of Rosenkrantz [25], and m is the number of region.

On the other hand, as regards the heuristic information, the value is always equal to the inverse of the distance between two regions as in equation 4.4, as this parameter does not update unlike the pheromone, that has a local and a global updating rule.

$$\forall j,k \in [0,m], \quad \tau(j,k) = \frac{1}{m \times L} \tag{4.3}$$

$$\forall j, k \in [0, m], \quad \tau(j,k) = \frac{1}{D_{j,k}} \tag{4.4}$$

4a) Local Updating Rule: for the search process of each ant, there is a local update of the pheromone parameter as in 4.5 equation from [23], composed by an evaporation contribute ($0 \leq \rho \leq 1$), for each region, even those not visited, and a positive contribution linked to the τ_0 value if the region is visited. In general, since the value of $\tau(j,k)$ is greater than τ_0, at the end of the update the parameter decreases and the probability that other ants move from j to k decreases, choosing other paths and avoiding premature convergence.

$$\tau(j,k) = (1-\rho) \cdot \tau(j,k) + \rho \cdot \tau_0 \tag{4.5}$$

4b) Global Updating Rule: at the end of each iteration, after all the ant tours, the pheromone deposit is present through the global update. Also in this case the rule has two contributions as in equation 4.6 from [24], the one linked to evaporation, for which typically the value is identical to that of the local update ($\rho = 0.2$ decided after tests' results, as shown in Section 5.3) and the contribution linked to the length of the global best tour ($\Delta\tau$), with the intent to allocate more pheromone in shorter tours, as a proportional contribution to the inverse of the length. In any case, the evaporation takes place for each region while the additional contribution is only for the best global tour. The global update rule therefore allows the spread of "experiential" information, and allows the convergence of the ACS algorithm.

$$\tau(j,k) = (1-\rho) \cdot \tau(j,k) + \rho \cdot \Delta\tau(j,k) \quad with\ \Delta\tau\ as\ in\ 4.7 \tag{4.6}$$

$$\Delta\tau(j,k) = \begin{cases} D_{j,k}^{-1}, & if\ (j,k) \in global - best - tour \\ 0, & otherwise \end{cases} \tag{4.7}$$

Algorithm 2 Pseudo-code of Order Optimization [23]

Input: n Uav, m Region, l Ants

Output: Optimized tour for each UAV

for uav = 1 to n **do**
  $Region[i] \leftarrow$ assigned regions for U_i;
  Place N ants on the initial region of the UAV;
  $L \leftarrow$ tour length obtained from the nearest neighbour approach;
  $\forall i, j \in [0, m]$ $\tau(j,k) \leftarrow \frac{1}{m \times L}$, $\eta(j,k) \leftarrow \frac{1}{D_{j,k}}$;
  for iteration = 1 to ite **do**
    for ant = 1 to l **do**
      $R_j \leftarrow$ current region of ant;
      $\Pi_a \leftarrow Region[i]$ / $\{R_j\}$, $Order[ant] \leftarrow \{R_j\}$;
      while $\Pi_a \neq 0$ **do**
        $q \leftarrow$ randomly generated in [0,1];
        $R_k \leftarrow$ next region according Eq. 4.1;
        Move ant from R_i to R_k;
        Put R_k into $Order[ant]$;
        Remove R_k from Π_a;
        Update local pheromone according to Eq. 4.5
      end while
    end for
    Select the best tour solution;
    Update the global pheromone according to Eq. 4.6
  end for
end for

Chapter 5

Implementation and Results

5.1 Python

The entire ACS planning algorithm has been developed with the programming language "Python", thanks to the several open source libraries, and his adaptability with ROS (Robot Operating System), the most popular framework for robot programming and simulation. Python programming language design philosophy emphasizes code readability with the use of significant indentation and consistently ranks as one of the most popular programming languages.

In this Chapter some of the most important used libraries and, furthermore, a general review of the entire algorithm will be presented.

Some of the tools used are:

- **OpenCV:** the most important library of programming functions used. It stands for Open Source Computer Vision and is mainly aimed at real-time computer vision. It is tipically used in several fields, such as mobile robotics, 2D and 3D object detection feature toolkits, augmented reality etc. To support some of these area, OpenCV includes a statistical machine learning library that contains also Decision tree learning tool, Gradient boosting trees, Deep neural networks (DNN) etc. In this thesis it was useful for reading, convert and elaborate the initial image. Further information will be added in the next Section;

- **NumPy:** it is another important library for Python, it allows to work with vectors and matrix, necessary for working with a large amount of variables such as the pixel of the image, the number of cells, coordinates of points etc. It includes also a lot of tools to elaborate vectors and matrix, for operation and it is essential also to return variables that are input for others tools;
- **statistics and math:** used for mathematical operations;
- **random:** used to generate random integer values, for example the color code for different cells, or floating values, for the set of point in the Voronoi diagram generation;
- **Shapely.geometry:** is the library for some important geometric tools, including "Points", a specific variable with a tuple of two coordinates (x,y), "Polygon", exploited to create a polygon from a list of Points, its output is a tuple of points and has some important attributes as described in the next Section;
- **Matplotlib:** is the library that permits to have plot such as in MATLAB, plotting functions, geometries and to elaborate and save images. It has been used also for interactive plots and interactive button in plots;
- **gevoronoi:** is the library for the Voronoi diagram, it includes two most important function to create the region partition (described below) and to plot the different cells with different Color's, marking centroids, in the area with obstacles;
- **triangle:** it includes the Delaunay triangulation, with 5 possible parameters to improve the tessellation and, in this thesis, to avoid some of the principals issues linked to the presence of obstacles.

5.2 Tools Implementation

In this Section some Python tools will be studied in depth, moreover, while in the previous Sections and Chapters the general implementation scheme has been described, the final configuration of the algorithm includes some tools optimized both for the insertion of real map, obstacles and for UAVs.

Geometry

The **Shapely.geometry** library allows the use of *Polygons*, geometric variables that have a tuple of *Points* for their description. In addition to providing this type of variable, however, the library allows the definition of some characteristics and some important functions. Among these we find the possibility of immediately evaluating the area of the polygon with the *Polygon.area* function (useful when assigning the cells) and the centroids with the *Polygon.centroid* function. Furthermore, the polygon itself can be generated with holes inside, as in the presence of obstacles, by inserting further tuples of Points, as if they were polygons that drill the main geometry.

Voronoi

The **geovoronoi** library allows to generate the corresponding Voronoi diagram using the *voronoi_regions_from_coords* function, which accepts as input the list of generating points of the cells, and the entire region (also with no-fly-zone) to be divided, in the form of a polygon. Unlike other tools present for the Voronoi diagram, this is the only library that allows to use limited perforated maps, instead of generic unlimited Cartesian spaces and without the constraint of obstacles, resulting in a powerful means for the tessellation.

Triangulation

As described in Chapter 3, Delaunay triangulation is allowed by the *triangle* library. The *triangulate* function accepts as input a dictionary and a list of parameters in the form of a string.

The dictionary must include:

- the set of initial points that generate the map, i.e. the coordinates of the vertices of the obstacles and the image itself;
- the list of segments that join the vertices, in the form of a pair of numbers associated with the index order of the vertices in the first variable;

- a list of coordinates that correspond to the spaces to be left empty (the obstacles). If absent, no obstacles are present.

The usable parameters, on the other hand, can be expressed with letters and numbers, depending on the type of triangulation, those in the Table 5.1 are used.

The final output of function is a set of points, the old vertices with the addition of Steiner points for a better triangulation, used as input for the geovoronoi function.

Parameter	**Meaning**
”p”	Allows to use the dictionary as input
”q”	Allows to generate the conforming triangulation
”D”	Ensure all triangles are Delaunay
”a” + ”number”	Set the maximum cells area to ”number” value

Table 5.1: Parameters of *triangulate* function

User Interaction

For the more immediate and intuitive insertion of obstacles and maps, as well as the classic analytical method through code, it was decided to create two mouse callbacks, thus also being able to add real scenarios to the code. Specifically:

- A callback that receives as input the left click of the mouse in the position where you want to insert a UAV. The output is a dictionary with the [x, y] pair of coordinates for each drone.

- A callback that receives as input the left mouse clicks for each vertex of the obstacles, but the last vertex must also be defined by pressing the ”alt” key on the keyboard, with consequent closure of the polygon. The output is a Polygon type variable through the list of identified points.

The following images (Fig. 5.1-5.2) are an example of how obstacles can be defined for two different maps, the first near the UNESCO site of Castel del Monte, for which it was

imagined to fly over the entire natural area with the exception of the areas of the fortress and of the underlying inhabited structure, the second in the city of Bisceglie, for which it was decided to cover all the road areas, including the square, thus excluding the various buildings.

Figure 5.1: Original Map of Castel del Monte with UAV on the left and obstacles definition on the right.

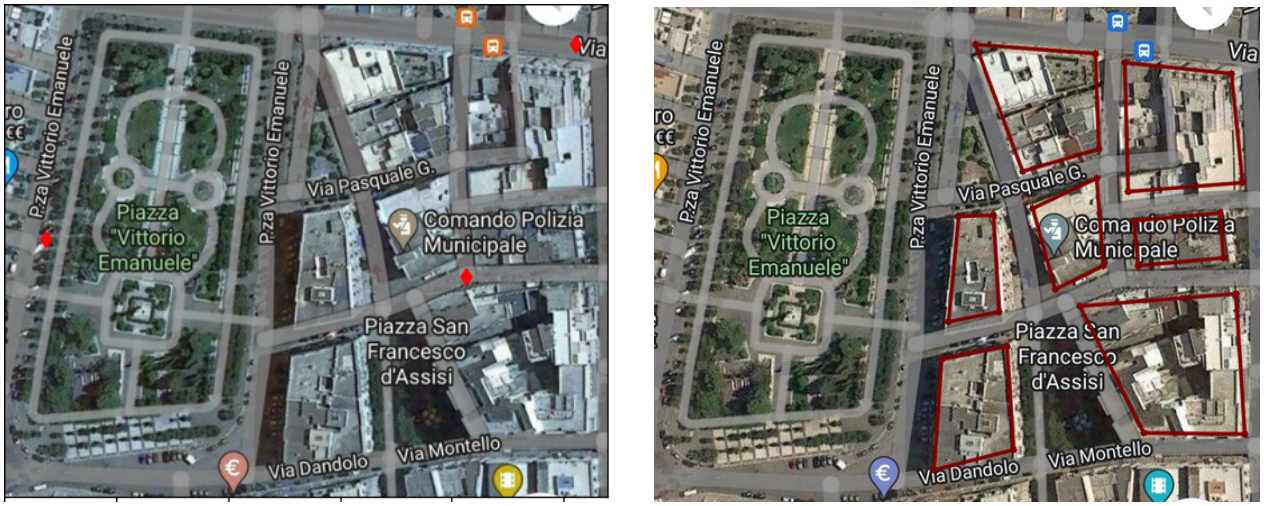


Figure 5.2: Original Map of Bisceglie with obstacles definition.

The definition of Voronoi diagram is in Figure 5.3.

Subsequently, there are also the different zones assigned to 3 drones in Figure 5.4, with different colors, without distinction of the individual cells but as if they were a single area.

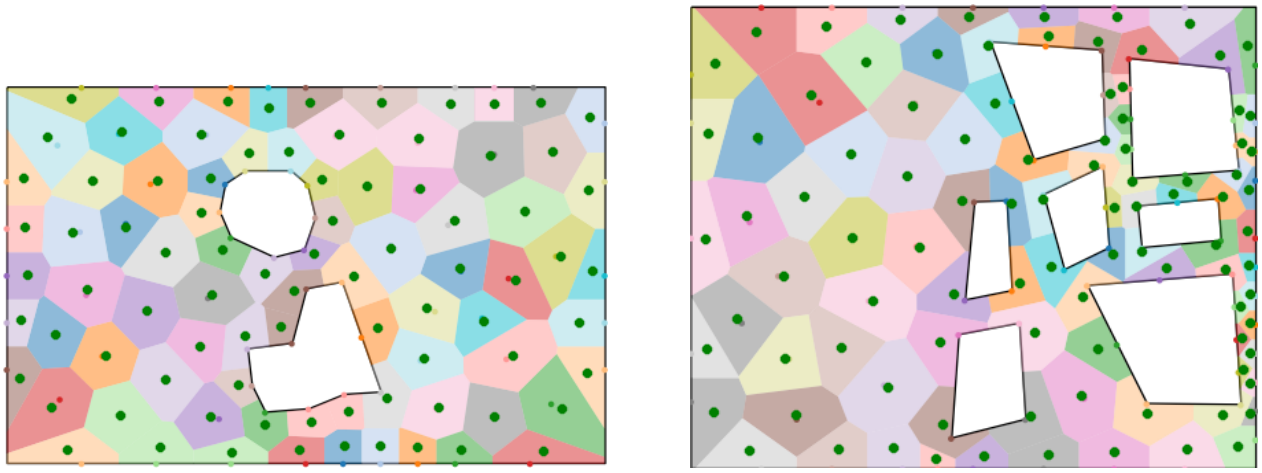


Figure 5.3: Voronoi diagram for Castel del Monte and Bisceglie.

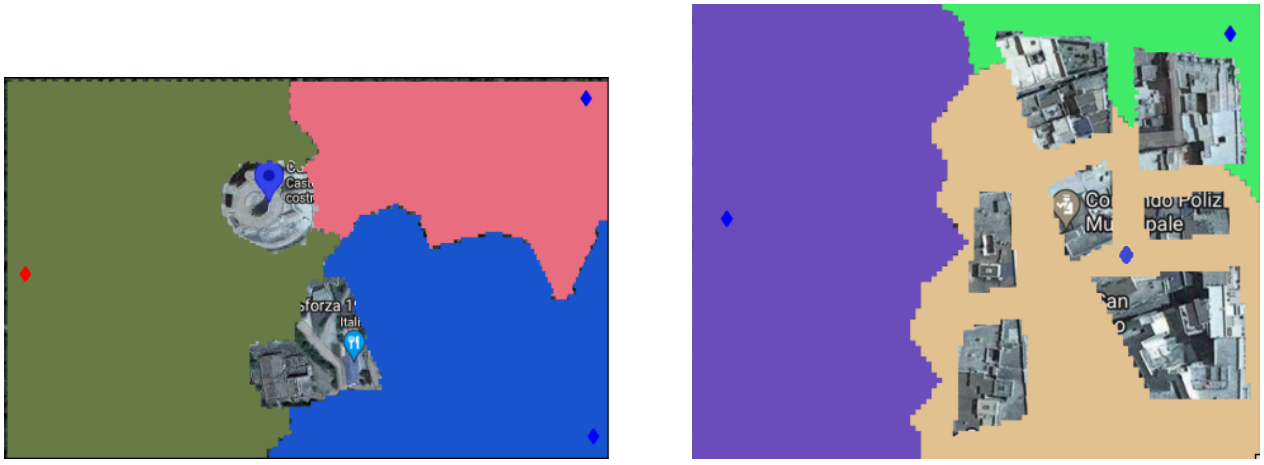


Figure 5.4: Subdivision between 3 UAVs.

5.3 Final Results

To substantiate the chosen algorithm feature, different tests were conducted at the end of the work, observing a correspondence between the expectations and the final results of the simulations. In particular, the focus is on the comparison between the two different subdivisions (Chapter 3), the choice of the parameter relating to the area of the cells in the Delaunay triangulation (Section 5.2) and finally the parameters that regulate the optimization through the Ant Colony Algorithm (Chapter 4).

5.3.1 Boustrophedon Decomposition vs Voronoi Diagram

A first study was carried out on the variation of two main parameters, the number of UAVs present in the map and the number of obstacles, to compare the two different methods of subdivision into cells of the area to be covered. In particular, 5 maps were randomly generated as in Figure 5.5, to observe the behavior for more than one available solution. For all the maps from 1 to 4 obstacles and from 2 to 5 drones were chosen, all UAVs with the same field of view equal to one pixel (equivalent to one meter), for a higher resolution coverage of the image but, in any case, the FOV does not affect the overall behaviour if it is identical for all drones. Furthemore, the drone had same cruise speed (m/s), so as not to alter the results with more performing drones than others. The flight times of the various drones were then evaluated ($\frac{Cell\ Surface}{FOV \times Speed}$), sampling the maximum, equal to the time needed to cover the entire area.

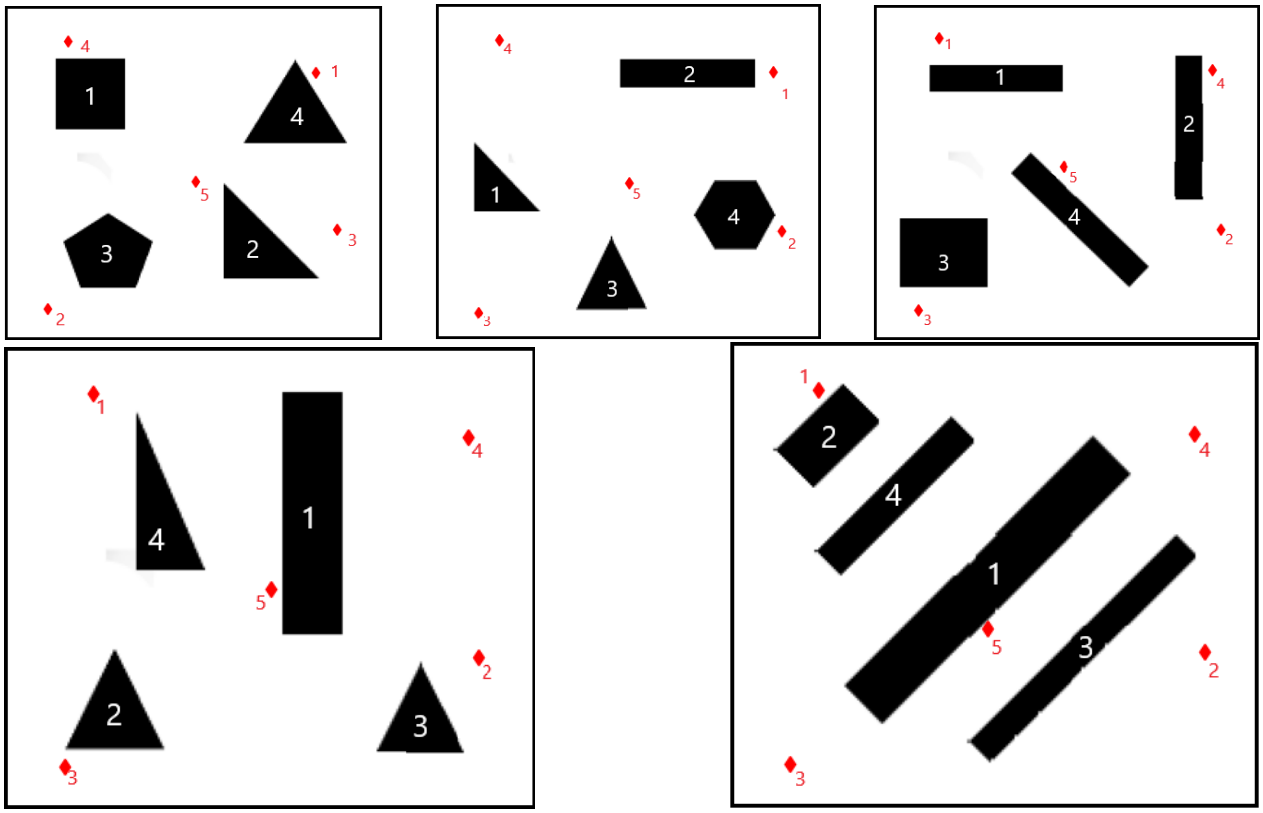


Figure 5.5: Original Map for testing the two decomposition. White/red number for addition order of obstacles/UAV.

The results for the first map show a substantial improvement in the algorithm by choosing the subdivision using the Voronoi diagram (choosing the conformed constrained Delaunay triangulation with area parameter). In fact, as visible in the graphs of Figure 5.6, observing the trend as the obstacles present in the map increase, therefore with a constant number of UAVs for each graph, the subdivision using Voronoi diagram (VD) presents lower total time of coverage, although the number of cells of this is much greater than the Boustrophedon decomposition (BCD), this is because, in general, the cells of the BCD are very variable on the surface, and some drones find themselves covering much larger areas than others. However, for both methods, as the obstacles increase, the total time decreases, as the total area to be covered is less.

Finally, the sum of the times of UAVs in the BCD is sometimes less, as the passages between cells are much less, but the distribution between drones is uneven, so the total time of the mission is still higher.

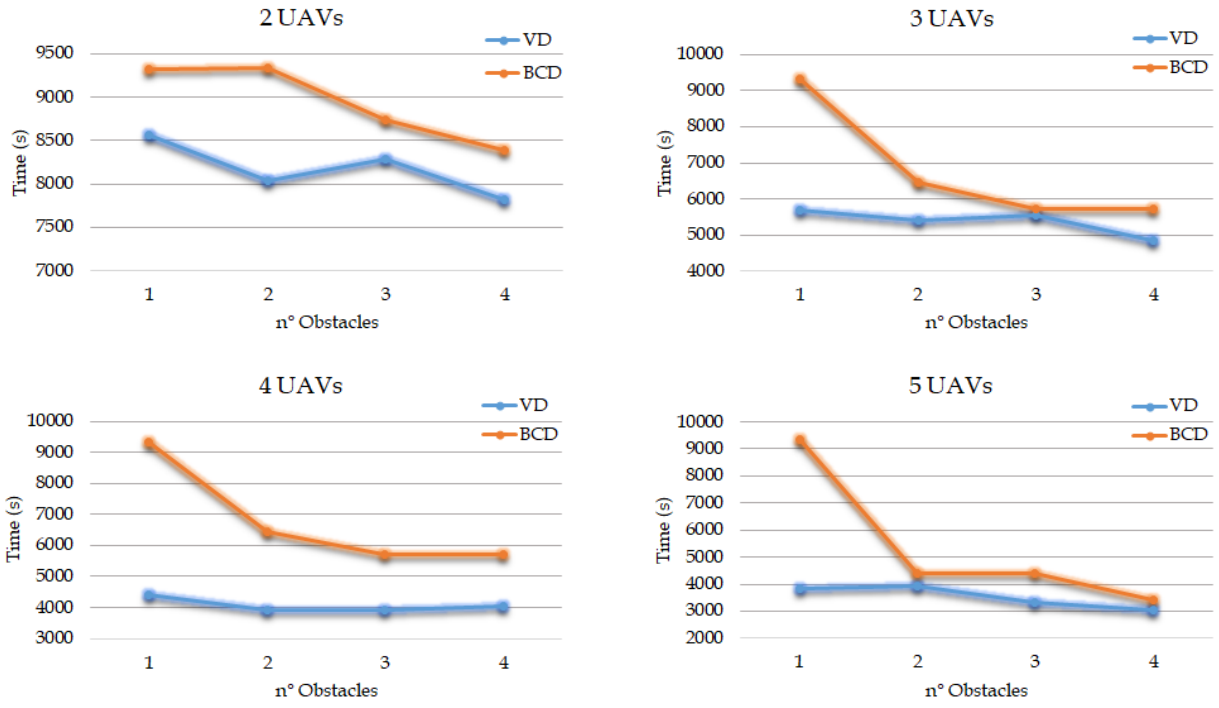


Figure 5.6: Elapsed time for coverage varying number of obstacles.

An important anomalous behavior occurs for few drones and few obstacles in the BCD, because in some cases some drones are allocated in the same initial cell, so it happens that

only one of the two is allocated the region and the other drone remains inactive, increasing the total time, showing a wide deviation between the two graphs. Indeed, sometimes, some drones are totally unused due to this too broad subdivision such as in Figure 5.7 for the previous maps, making the inclusion of a multi-UAV system in the coverage vain to reduce total times.

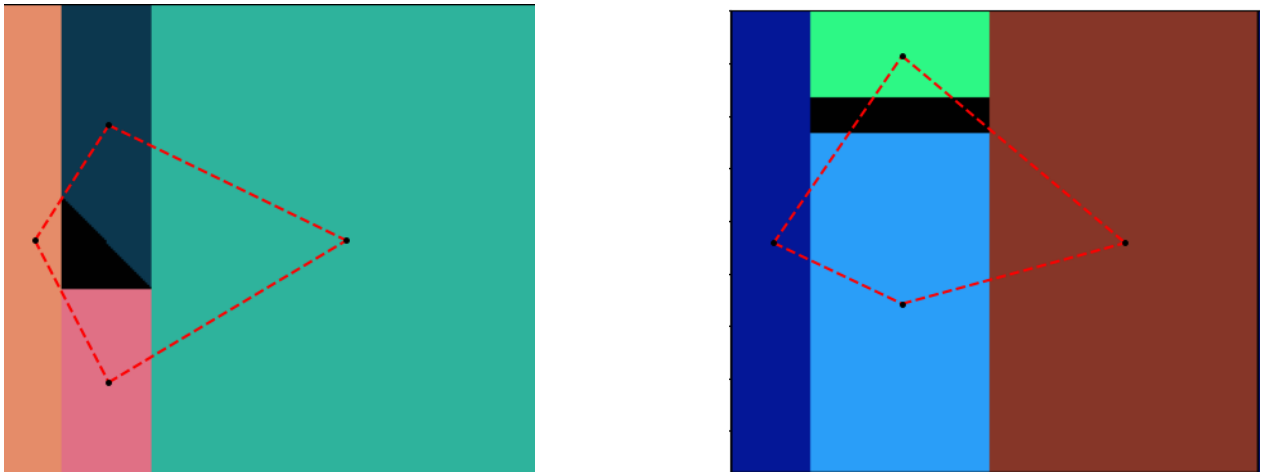


Figure 5.7: Abnormal cells for Boustrophedon Decomposition.

On the other hand, by analyzing the graph as the number of drones varies (Fig. 5.8), with constant number of obstacles, as desirable, it is observed a typical behavior of the VD which has reduced times increasing UAVs, justifying the choice of the multi-UAV system, even if obviously the increase must be limited by comparing the gain in time with the costs of purchasing and managing numerous drones. However, a sometimes constant trend is observed for the BCD, this is because, the very large cells, often incorporate the same UAVs that find themselves covering the same total surface, thus limiting the gain that occurs with the multidrone system, nullifying the subdivision in cells.

To corroborate the previous results, the trend of the same graphs was then observed (Fig. 5.10-5.11), including all the 5 maps generated, averaging the results of each. Generally the behavior is always the same:

- the total mission time decreases as the number of UAVs increases because each covers a smaller area (Fig. 5.12);

- the total mission time decreases as the number of obstacles increases because there is less area to cover (Fig. 5.12);
- the Voronoi diagram gives better results than the Boustrophedon decomposition diagram;
- with numerous UAVs and few obstacles the difference between the two is always remarkable.

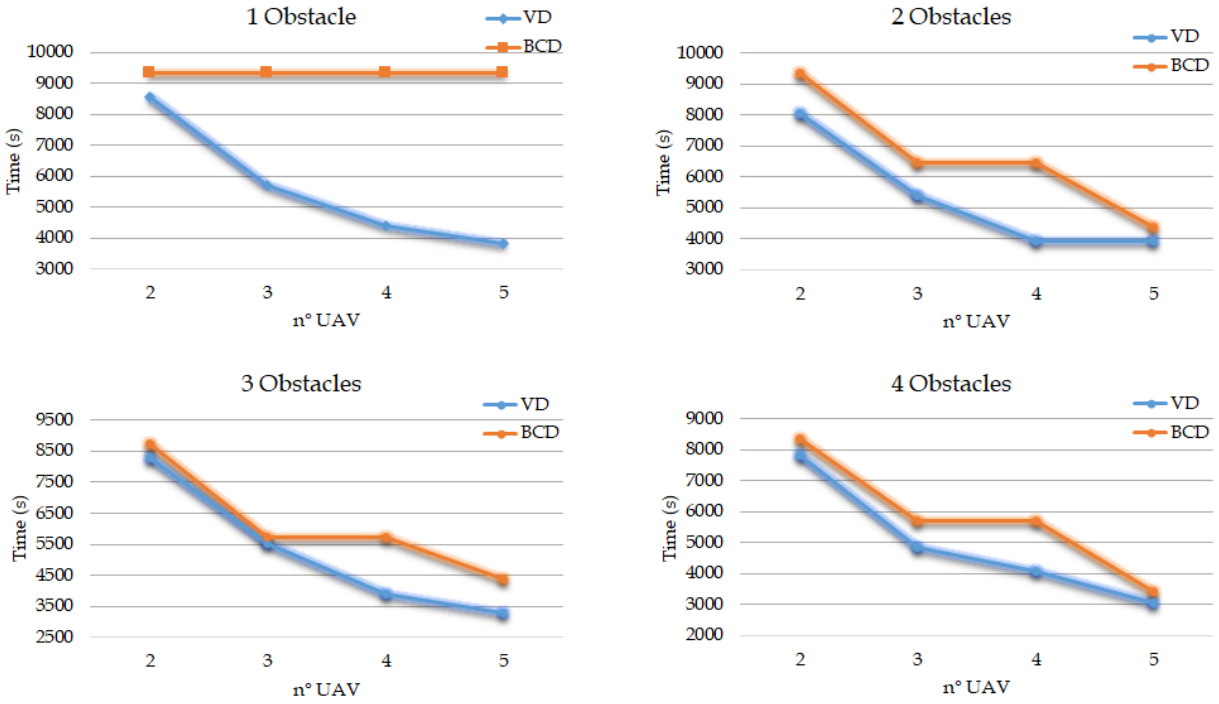


Figure 5.8: Elapsed time for coverage varying number of UAVs.

Furthermore, although the total number of BCD cells is always much lower (4-15 cells), and therefore the number of transitions between cells is lower, the total mission time is lower for the VD (60-170 cells) because they are better assigned.

Finally, the only anomalous behavior occurs for a few obstacles and only two UAVs, for which sometimes the VD is better and sometimes it is worse, depending on the variability of positioning of obstacles and UAVs, however it is an isolated case compared to different combinations and concerns a deprecated solution for this work.

Another very useful application concerns the selection of UAVs based on their characteristics. As described in the previous Chapter, the selection of the cells is made according to the FOV and the cruise speed of the drones; a drone with higher speed and wider scanning area is chosen more than another. In the examples shown in the Figure 5.9, using only the decomposition through Voronoi diagram as it is more efficient as presented above, it is observed that some drones cover larger areas than others, precisely because in reality they are able to cover larger areas in equal times compared to other drones less performing. This adaptability can be useful both in the case of different UAVs, of the same parent company but with subsequent developments, and in the case in which in some areas the drone cannot be used at its cruise speed but has some limitations related to regulations or different constraints. Further considerations can be made by introducing the altitude parameter, which can both link speed with the FOV, and be useful for differentiating drone flights, allowing to distance them vertically rather than horizontally.

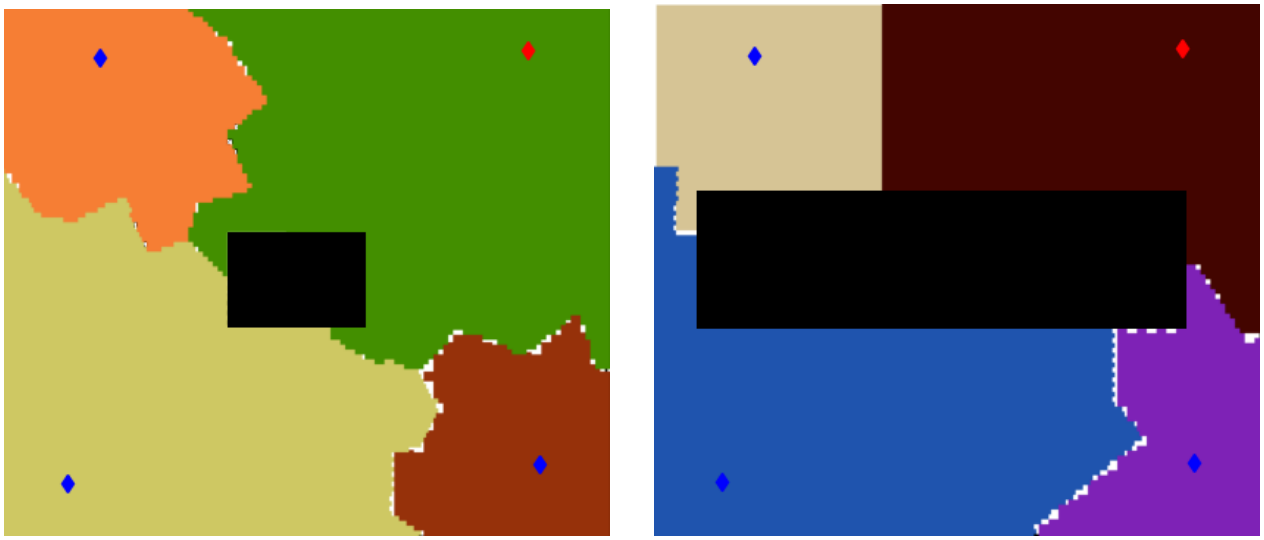


Figure 5.9: Region assignment with different FOV and cruise speed.

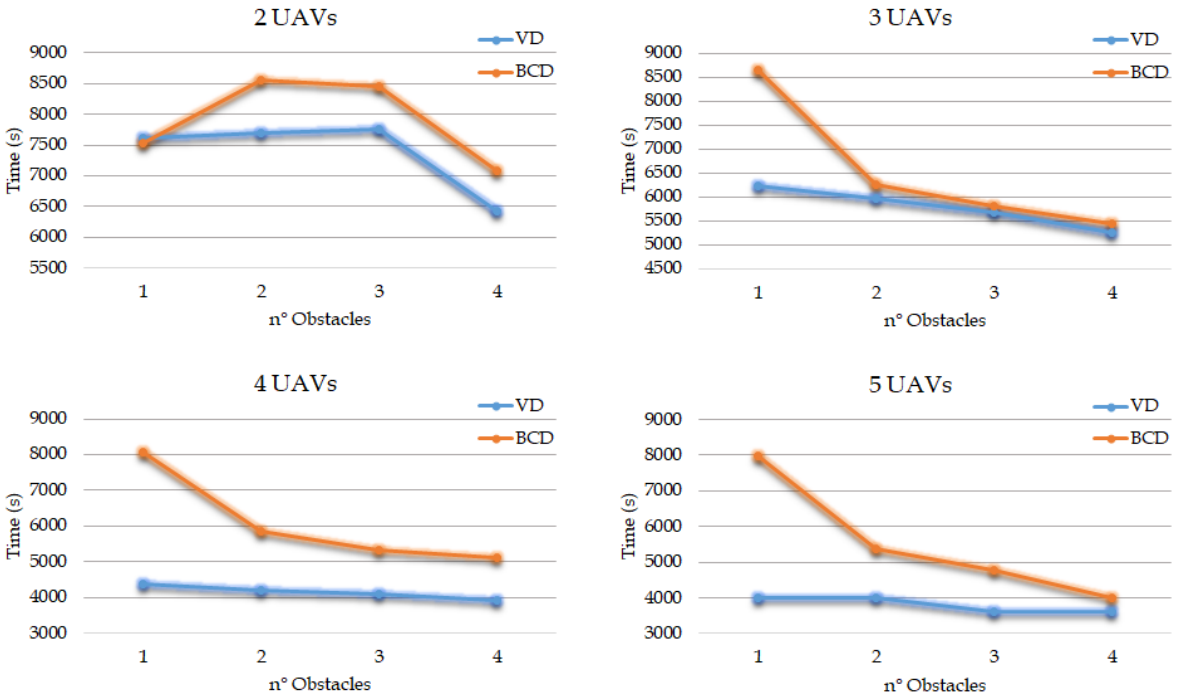


Figure 5.10: Elapsed time for coverage for 5 maps, varying number of obstacles.

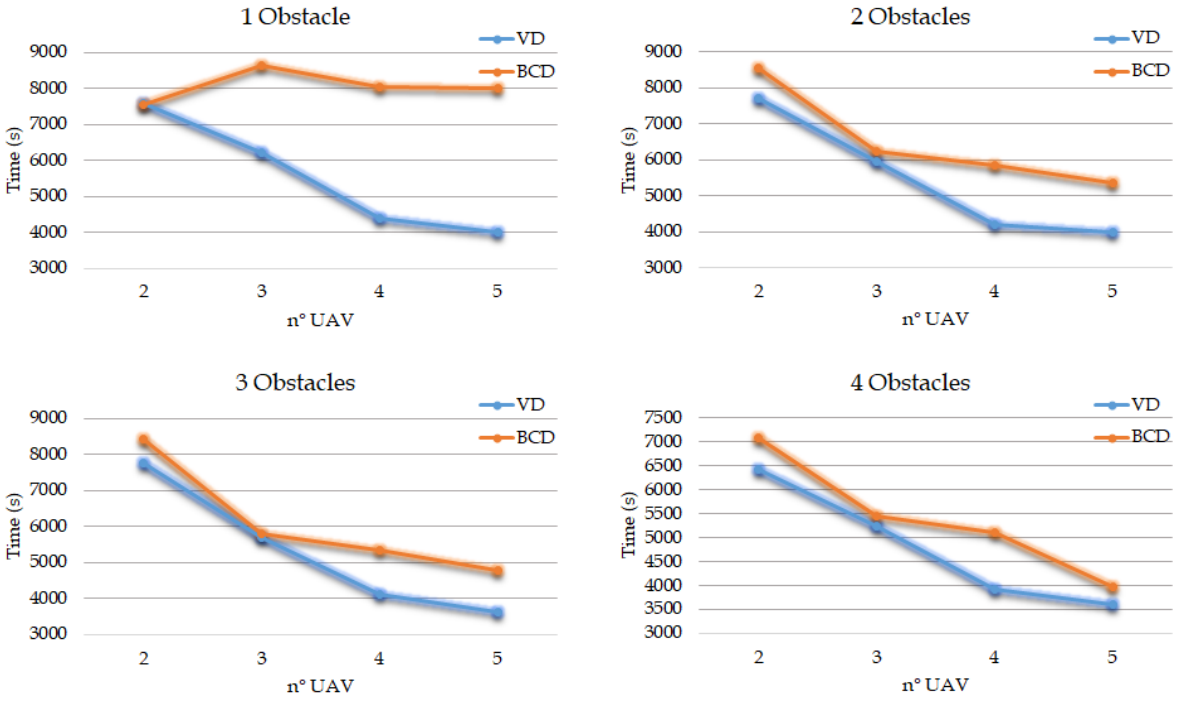


Figure 5.11: Elapsed time for coverage for 5 maps, varying number of UAVs.

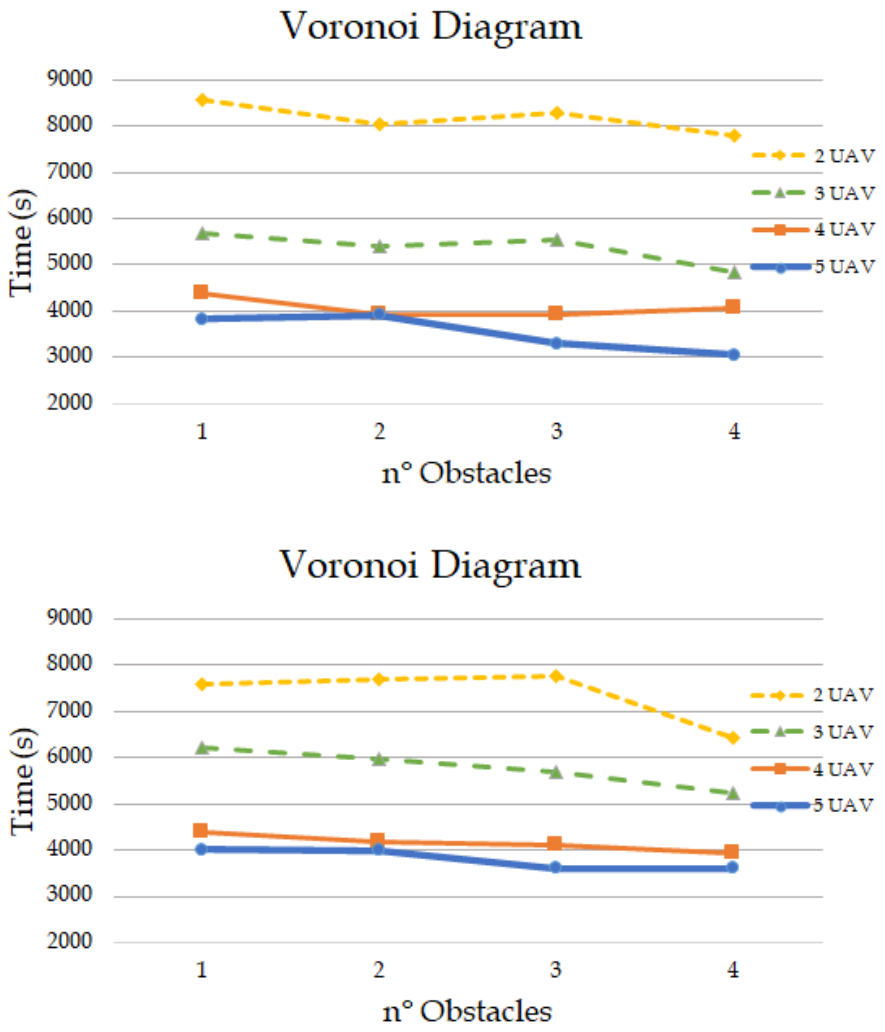


Figure 5.12: Voronoi diagram elapsed time for coverage summary.

Boustrophedon Decomposition	**Voronoi Diagram + Triangulation**
- Easier Computation	- Shorter mission time - Better subdivision for UAVs - All active UAVs
- Longer mission time - Some inactive UAVs - Worse subdivision	- Higher number of cells - Higher Computation

Table 5.2: Summary of BCD and VD comparison.

5.3.2 Delaunay Triangulation Optimization

For the definition of a correct parameter "a" of the Python *triangulate* function, the initially focus has been on a value proportional to the area of the largest obstacle, and in the case of a map without obstacles, proportional to the entire area, both to have more regions in case of 0 obstacles, and, above all, to avoid that some cells were difficult to cover with a drone, as described widely in Chapter 3.

However, further analysis can be conducted for this parameter in relation to a better optimization of the subdivision. To highlight which value was optimal, some tests were conducted, for two different maps, with 2 and 4 identical drones, depending on the parameter itself. Eight different values were chosen, all proportional to the area of the largest obstacle "a_{obst}" by means of a multiplicative factor, observing the distribution of the scanning times of the different drones, the standard deviation of these, the total time for each drone and the number of cells. In Figure 5.13-5.14 some triangulation and Voronoi diagrams are presented as the parameter varies, for the two different maps.

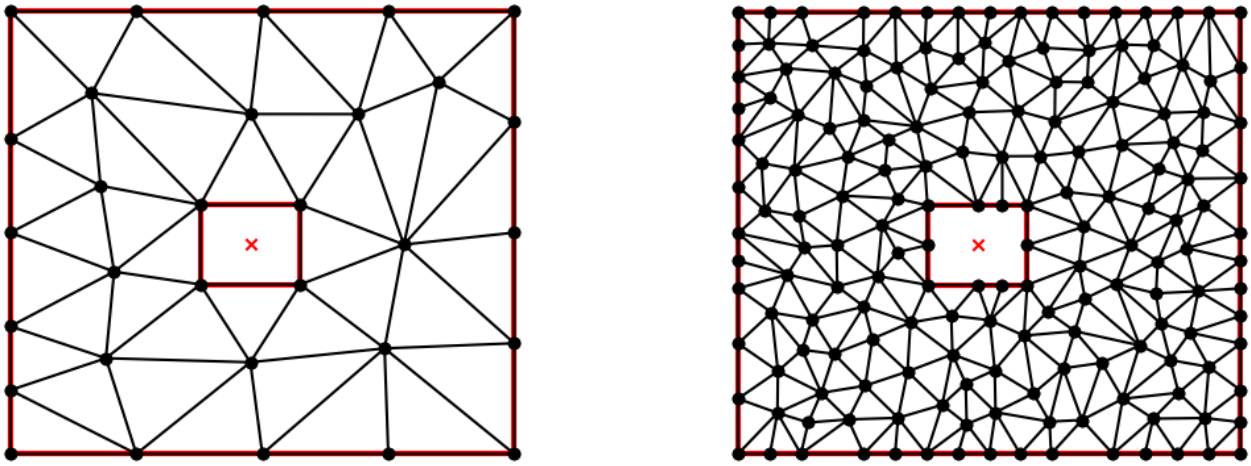


Figure 5.13: Delaunay Triangulation for "a_{obst}" and "$a_{obst}/8$".

As shown by the next graphs, the trends for the variables are as follows:

- the **average scanning time** for the various drones, apart from an initial irregular trend, decreases as the parameter decreases, because the subdivision is more equitable (Fig. 5.15);

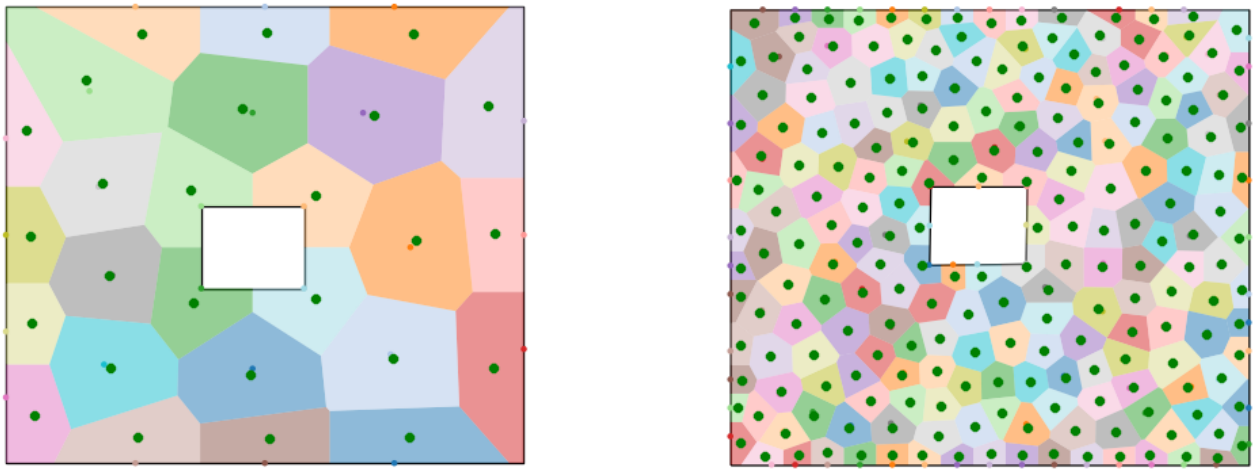


Figure 5.14: Voronoi diagram for "a_{obst}" and "$a_{obst}/8$".

- the **standard deviation** decreases as the parameter decreases, this is because the subdivision is more equal and the various drones begin to have the same scanning area approximately, however the null value is never reached because the cells are not all identical (Fig. 5.16);
- the **number of cells** increases as the parameter decreases, because each cell has a smaller area constraint (Fig. 5.17);
- the **total time of mission** (scan + fly between cells) for each UAV increases as the parameter decreases, because with equal scanning area, there are more cells and more paths through them (Fig. 5.18).

What can be deduced from the trends is that, choosing a parameter as small as possible, perhaps even proportional to the area of the smallest obstacle, optimizes scanning times and assigns the same total area to the drones, however, with very small cells the total time of the mission get worse, sometimes considerably, so it makes no sense to exasperate the choice of the parameter. In reality, this behaviour for small parameters occurs because the focus of the thesis is mainly that of optimizing the subdivision and the order of visit of the cells, rather than the travel path in the single cell. Finally, a value that can optimize this work for the area parameter is "$a_{obst}/4$", that is a quarter of the area of the largest obstacle, in order to have a low dispersion of the data but in any case a not high number of cells.

"a" parameter	Mean Scanning Time	Standard Deviation	Number of Cells	Total Mission Time
↓ Decrease	↓ Decrease	↓ Decrease	↑ Increase	↑ Increase

Table 5.3: Summary trends varying "a" parameter

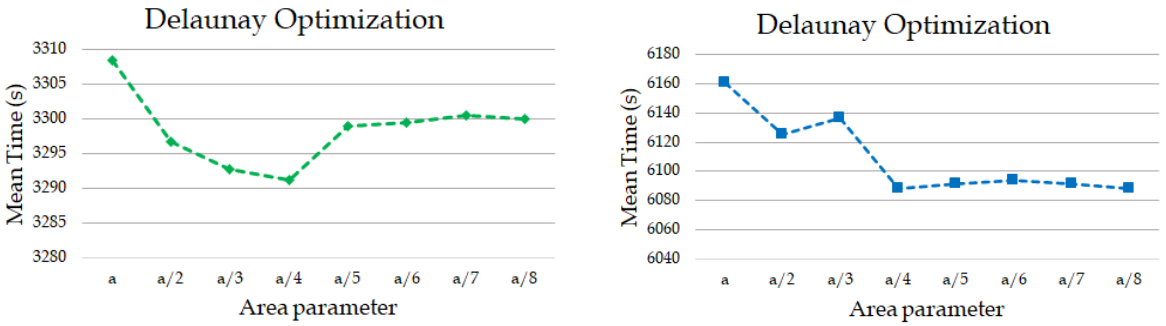


Figure 5.15: Mean time for each UAV for two maps.

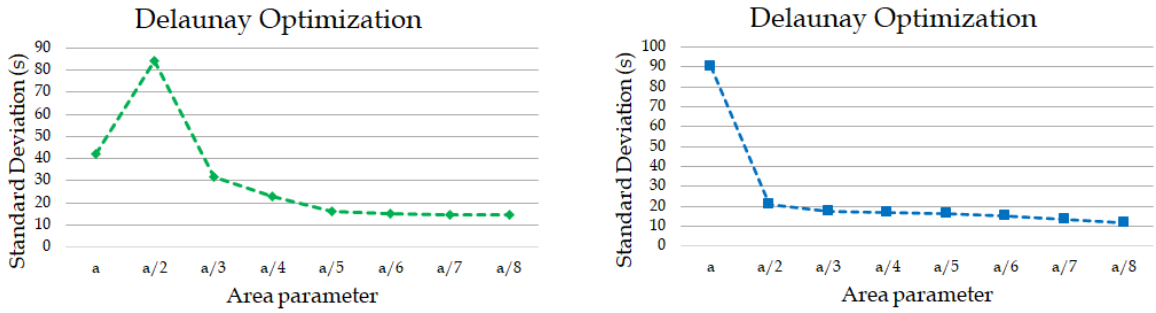


Figure 5.16: Standard Deviation for two maps.

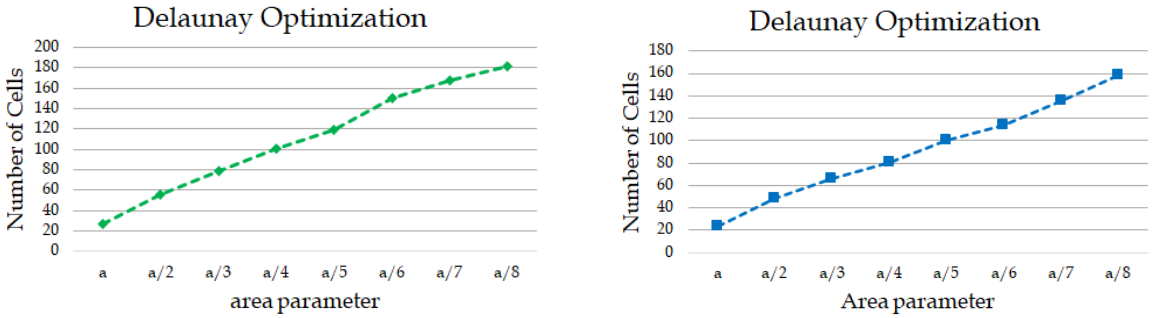


Figure 5.17: Number of cells for two maps.

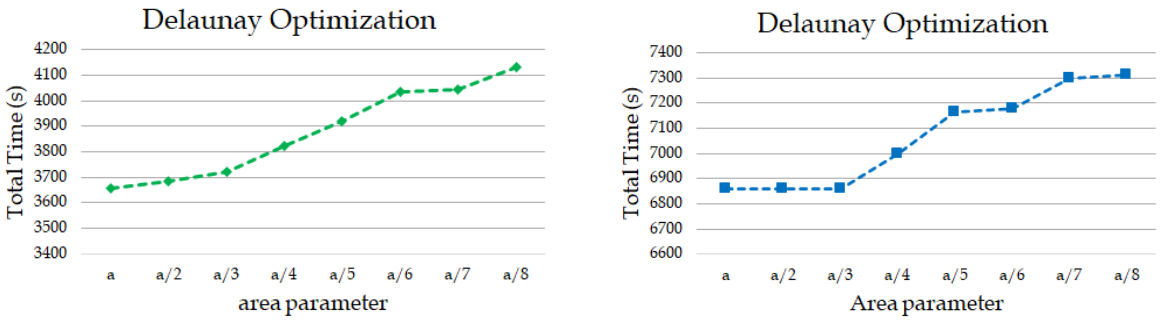


Figure 5.18: Total time for two maps.

5.3.3 Ant Colony Algorithm's Parameters

As for the optimization through Ant Colony, the main focus concerns the choice of the numerous parameters within it, to obtain a satisfactory result in terms of time, without however arriving at premature convergence of the method. In particular, the following parameters were analyzed by carrying out 10 tests for each single configuration, thus obtaining an average trend of the algorithm which, including a semi-random characteristic, does not always converge after the same number of iterations. The parameters chosen were:

- *Number of Ants*: is the number of ants (drones) used to test the different tours; after each iteration, as extensively explained in the previous Chapter, the method performs a global update until the tours of all the ant converges; in tests it changes from 2 to 7.

- *Evaporation Ratio "ρ"*: is the parameter that allows to update the pheromone value of each cell according to whether a path is chosen or not. The higher the parameter, the more the pheromone evaporates, so it will be felt less by subsequent ants; in tests it changes from 0.1 to 0.5.

- *α parameter*: is the exponent present in formula 4.1, weighs the contribution of the pheromone factor within the exploration of the ants. In literature it has a value between 0 and 1, the smaller it is and the less the pheromone has power, therefore the collaboration contribution between the different ants is weakened; in tests it changes from 0.5 to 1.0.

- *q0 threshold*: it is the threshold that distinguishes the two different behaviors within

the State Transition Rule (Eq. 4.1) discussed in the previous Chapter; if the random generated value is lower than the q0 threshold, the pheromone and heuristics information are exploited ("exploitation"), if instead the value is higher, the "exploration" is random according to the Random Wheel Rule (Eq. 4.2); in tests it changes from 0.7 to 1.0.

Overall, 10 tests were conducted for each of the 35 configurations, for a total of 350 simulations using the algorithm. The four parameters have been varied as in the Table 5.4, avoiding configurations that intuitively would have led to non-convergence of the method or to very poorly performing solutions; moreover, the maximum number of iterations was set to 1000 to avoid too long cycles.

Parameter	**Initial Value**	**Final Value**	**Step**
Number of Ants	2	7	1
Evaporation Ratio	0.1	0.5	0.1
α	0.0	1.0	0.5
q0	0.5	1.0	0.2

Table 5.4: Choice of parameters for the 35 configuration.

The map chosen is the one in the Figure 5.19, with the presence of 3 drones inside which, whatever the configuration tested, always covered the same colored area, this is because altering the parameters in question does not affect the assignment of the regions (dependent from geometric and physical parameters), but only the convergence of the sorting method of visiting the regions themselves.

Number of Ants

The first parameter chosen for the variation was the number of ants, in particular from 2 to 7 ants for the chosen map where there are 103 regions to be placed and the three drones have respectively 33, 37 and 33 cells to cover. As mentioned, 10 tests were conducted for each value of the parameter, and the average value of the number of iterations necessary to arrive at convergence was sampled. The remaining parameters have been kept constant to

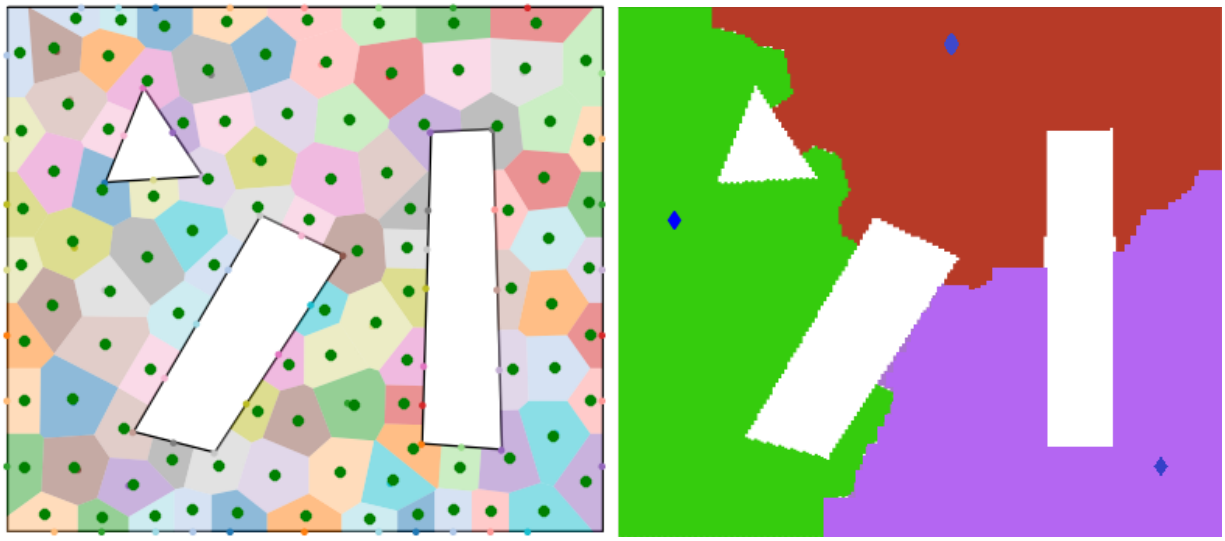


Figure 5.19: Voronoi diagram and region coverage for the selected map.

the values shown in the Table 5.5. The number of iterations necessary is increasing with the number of ants chosen (Fig. 5.20), this is because, obviously, more ants must reach the convergence of the path, however it makes no sense to observe only this value to judge which is the final configuration to choose, but also the time mission of the drones that the specific convergence leads to have.

	Number of Ants	**Evaporation Ratio**	**α parameter**	**q0 threshold**
Value	2:7	0.25	1.0	0.9

Table 5.5: Selected Parameters for different configurations of Number of Ants.

It can be observed that all the configurations with the highest number of ants find the same path (same total mission time), which is more efficient in terms of time than those with a low number of ants (fig: 5.21), this is because the low number of iterations actually converges the solution at a higher value, dictated by the anticipated convergence in the presence of too few ants.

Even observing the ten different tests (fig: 5.22-5.23), it is observed that for a number of ants greater than 4 the solutions are approximately the same, while for a number less than 4 there are some worse solutions.

Finally, since too high number of ants is still overabundant (5 ants reach the same solution

of 6 or 7 with low number of iteration), and too low number is not congenial, it was decided to conduct the subsequent tests only for a limited range of variability of this parameter, trying to obtain the best solution.

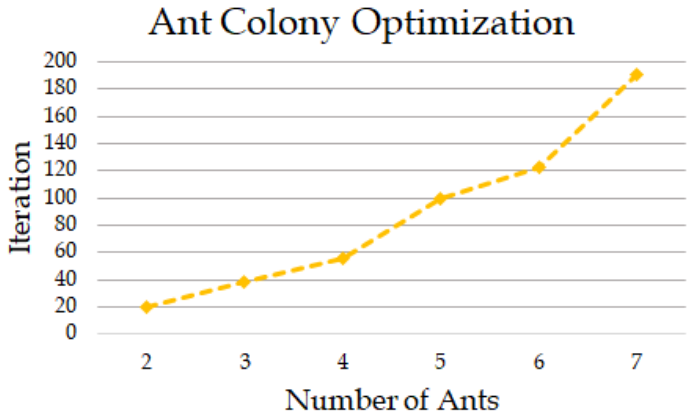


Figure 5.20: Average number of iteration varying Number of Ants.

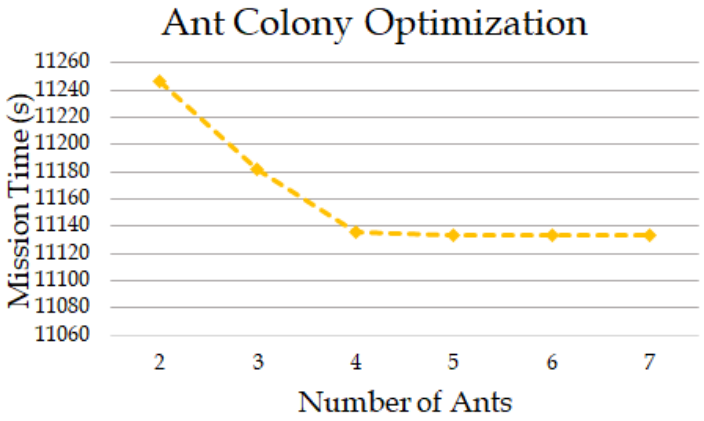


Figure 5.21: Average Total Time varying Number of Ants.

Evaporation Ratio

After the first parameter, the Evaporation Ratio was varied to see how the algorithm responded. The values chosen were from 0.1 to 0.5, excluding the null value because it would have meant the absence of evaporation, and excluding the higher values following some tests conducted. Furthermore, it was decided to sample the results for 3 different numbers of ants

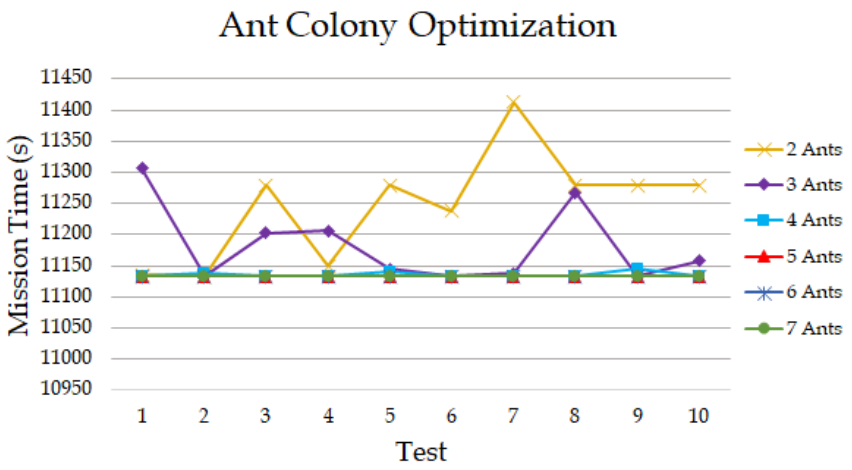


Figure 5.22: Total Mission Time for each test varying Number of Ants.

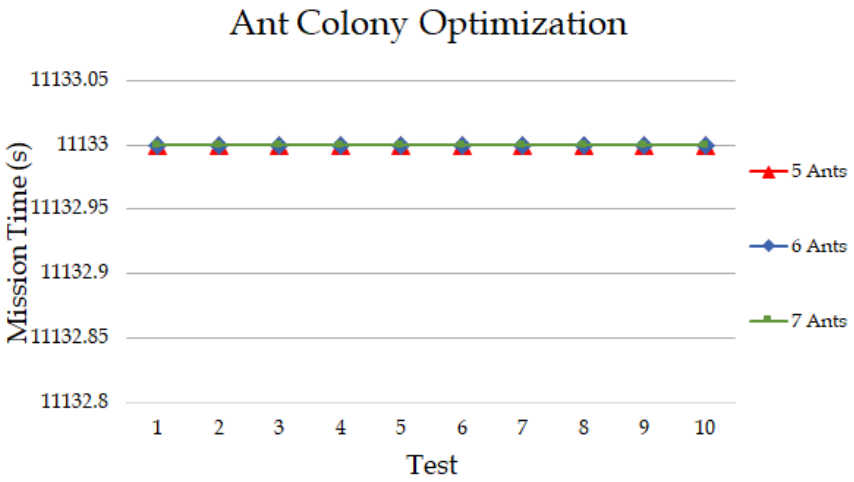


Figure 5.23: Total Mission Time for each test varying Number of Ants.

as mentioned in the previous Section. The remaining parameters have been kept constant to the values shown in the Table 5.6.

	Number of Ants	**Evaporation Ratio**	**α parameter**	**q0 threshold**
Value	3:5	0.1:0.5	1.0	0.9

Table 5.6: Selected Parameters for different configurations of Evaporation Ratio.

The behavior when ρ varies is approximately the same for each number of ants, e.g. for increasing values, the number of iterations necessary for the convergence of the ten tests

increases (it makes no sense to test configurations with too many ants because they will get too long), until there are solutions that, in some tests, exceed the limit of 1000 iterations (samples circled in red in the graphs of the figures 5.24-5.25-5.26), instead very small values lead to convergence very early with a low number of iterations. For 4 and 5 ants, and evaporation ratio greater than 0.3, all tests have at least one drone that has no convergence, as well as for 3 ants and values greater than 0.4.

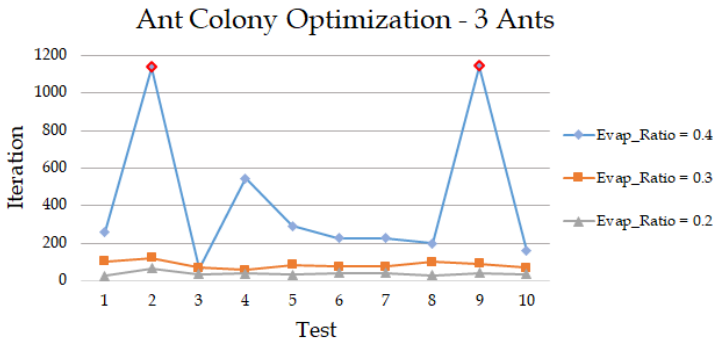


Figure 5.24: Number of iteration for each test varying ρ.

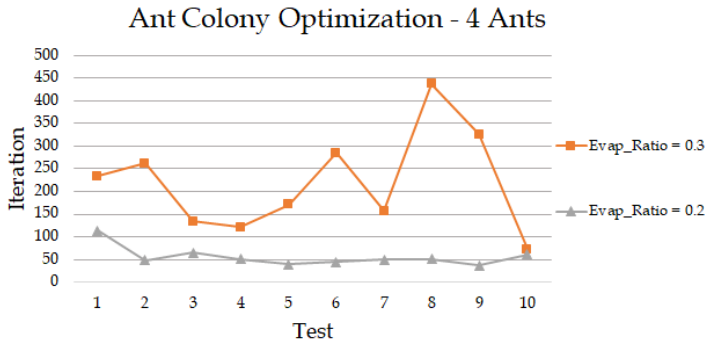


Figure 5.25: Number of iteration for each test varying ρ.

Then, two cases for each number of ants were selected, those with low iterations, and for these, as in the previous case, the mission time was observed (Fig. 5.27). Also in this case some solutions give too early convergence, with longer mission times than the optimal solution. Finally, a solution was selected for each Number of Ant value, the one with correct mission times, and it was then chosen to continue the tests with these fixed values, although one appears to have a higher output variability, as shown in the Figure 5.28.

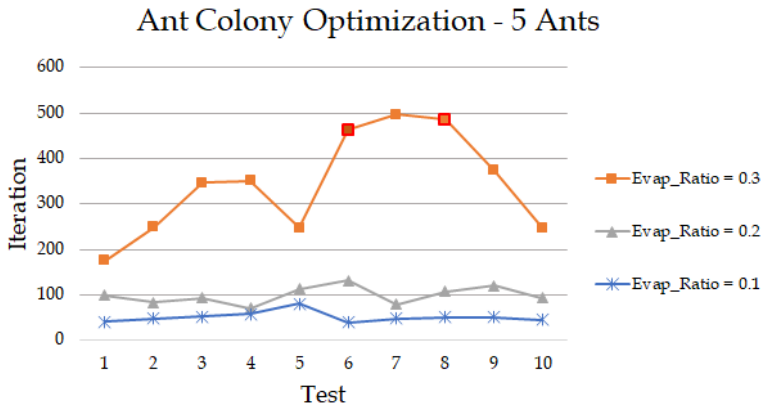


Figure 5.26: Number of iteration for each test varying ρ.

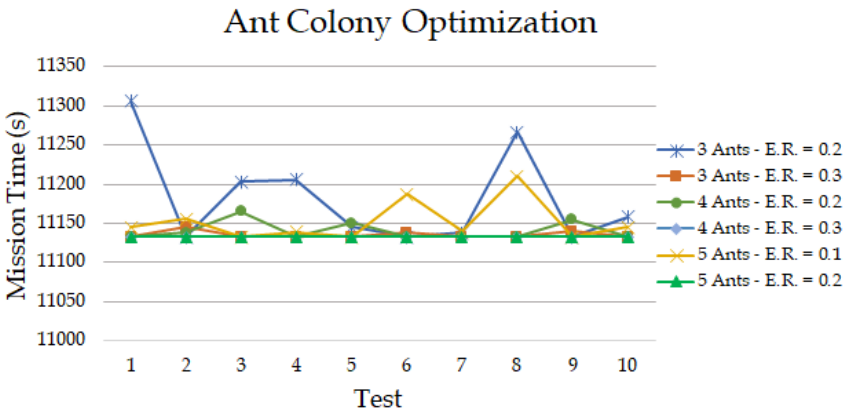


Figure 5.27: Total Mission Time for each test varying Evaporation Ratio.

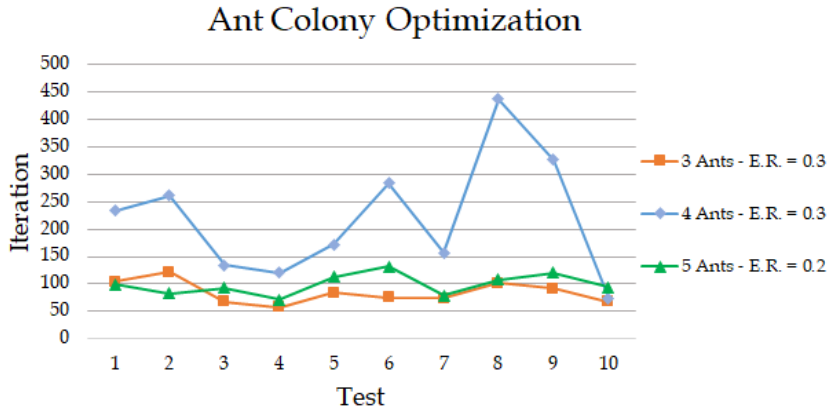


Figure 5.28: Number of iteration for each test for selected solution.

α parameter

The third parameter chosen is α, the exponent that provides the contribution of the pheromone information within the State Transition Rule. The remaining parameters have been kept constant to the values shown in the Table 5.7.

	Number of Ants	**Evaporation Ratio**	**α parameter**	**q0 threshold**
Value	3:5	0.2:0.3	0.0/0.5/1.0	0.9

Table 5.7: Selected Parameters for different configurations of α.

For each selected solution, if α is null, the iterations are higher than the set threshold (acceptable iterations if less than 1000), this is because the ants act as if one does not affect the other, a fundamental characteristic both in nature and in the algorithm for the convergence of paths. If instead the parameter is slightly lower (alpha = 0.5), the number of iterations is higher but still acceptable, and sometimes, as in the case of 3 ants, it is preferable to that of $\alpha = 1.0$ because a slower convergence leads to optimal solutions (Fig. 5.29-5.30-5.31). In fact, by finally viewing the mission times graph, for what seem to be the best solutions, the curves flatten out to the minimum value (Fig. 5.32). However, the final choice is for the number of ants equal to 5, evaporation ration equal to 0.2 and alpha equal to 1.0, the most stable solution for the 10 tests as well as with fewer iterations (Fig. 5.33).

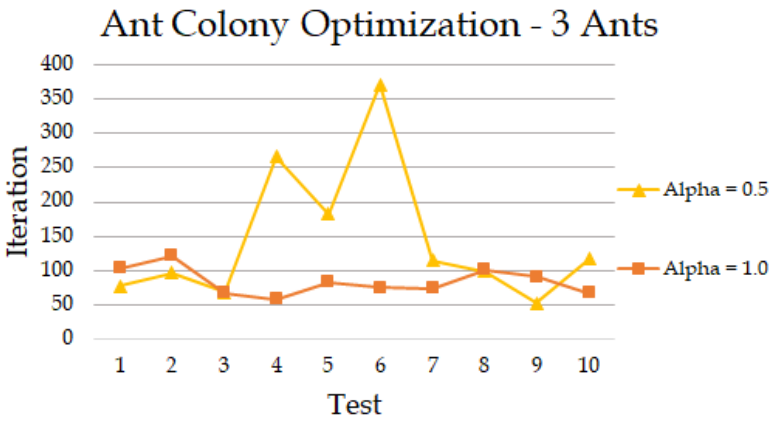


Figure 5.29: Number of iteration for each test varying α.

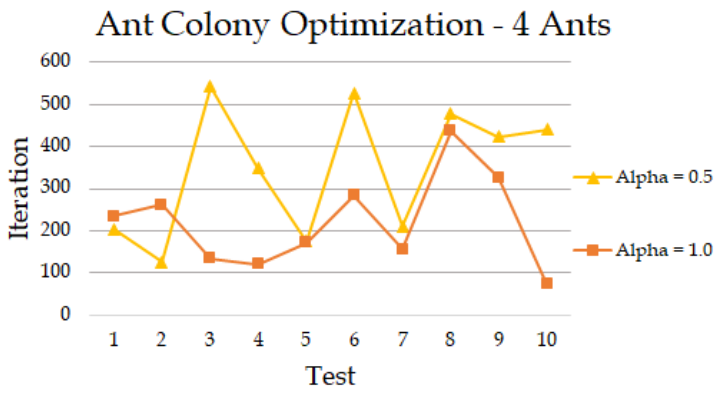


Figure 5.30: Number of iteration for each test varying α.

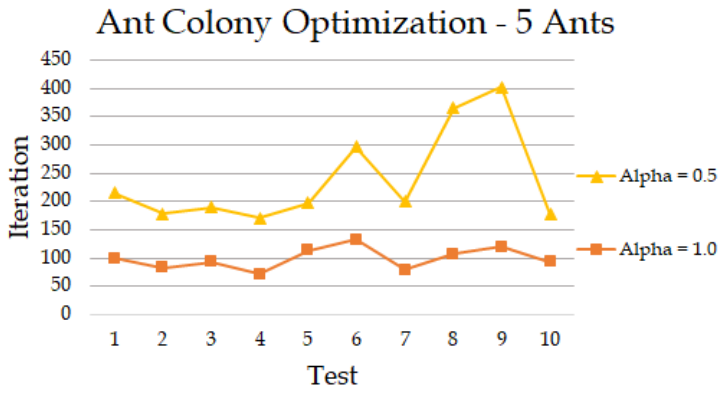


Figure 5.31: Number of iteration for each test varying α.

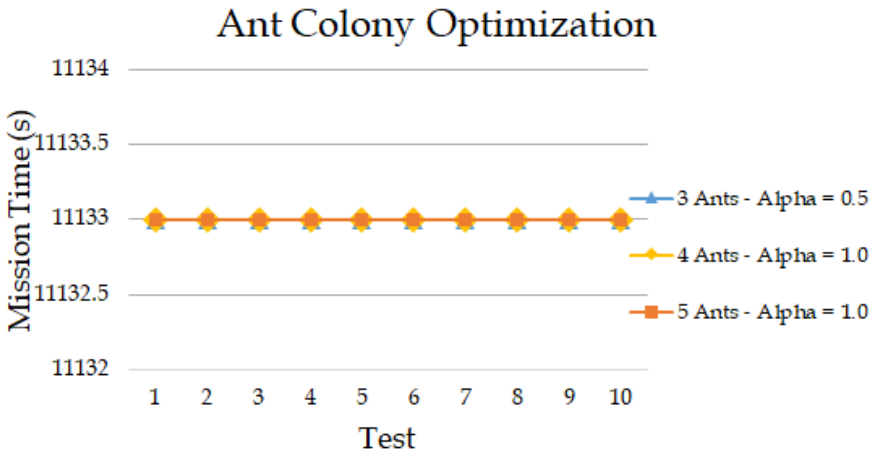


Figure 5.32: Total Mission time for each test varying α.

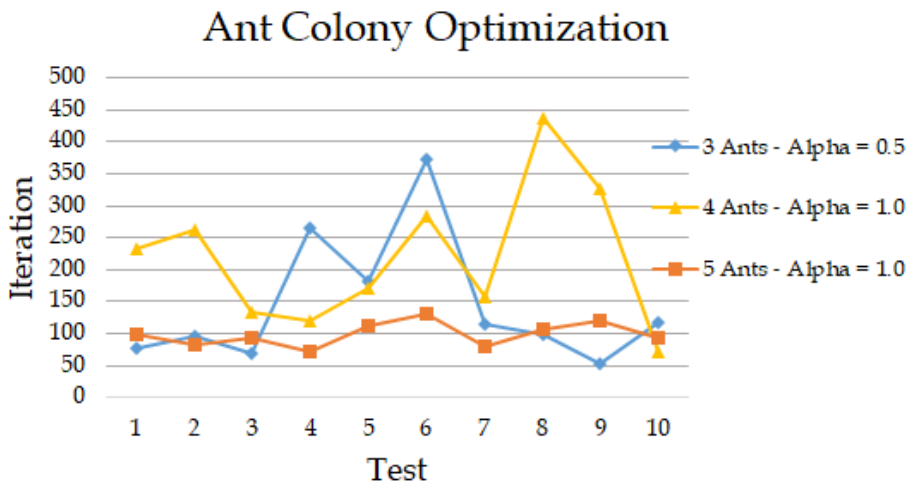


Figure 5.33: Number of iteration for each test for selected solution.

q0 threshold

The last parameter chosen is q0, the threshold value that guides the exploration of ants according to the State Transition Rule. The chosen values were of 0.7, 0.9 and 1.0, which, if exceeded by the random value generated during the optimization, lead to the selection through the Roulette Wheel as described in the Chapter 4. Therefore, a lower value favors random paths, useful for avoiding premature convergence, a higher one rather lead always to the same fixed paths. The remaining parameters have been kept constant to the values shown in the Table 5.8.

	Number of Ants	**Evaporation Ratio**	**α parameter**	**q0 threshold**
Value	5	0.2	1.0	0.7/0.9/1.0

Table 5.8: Selected Parameters for different configurations of q0.

The behaviour for different numbers of ants is very similar, however the final configuration of 5 ants has already been chosen so only these graphs are plotted (Fig. 5.34). For q0 smaller, the path is too random and the number of iterations increases considerably, so it should be avoided, instead for q0 maximum, the paths are too quickly identical and the solutions sometimes do not converge to the optimum (best order in Fig. 5.36), therefore, by viewing also the graph of the total mission time, it was decided to maintain the value of q0 = 0.9 for

the final configuration (Fig. 5.35).

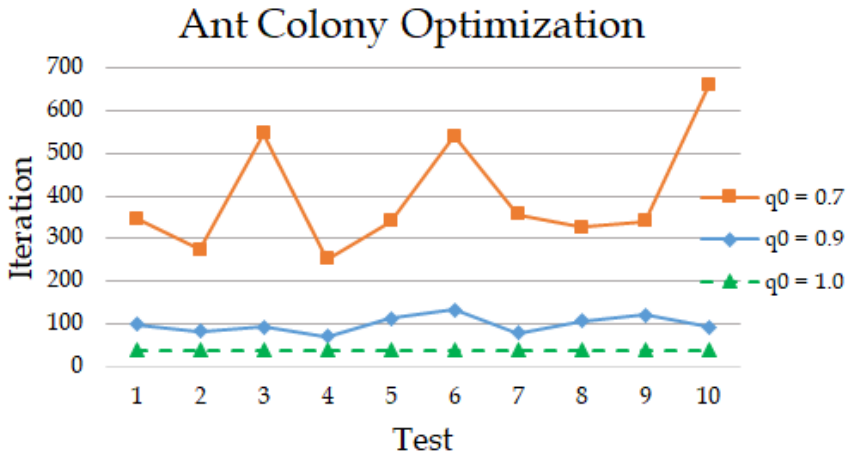


Figure 5.34: Number of iteration for each test varying q0 threshold.

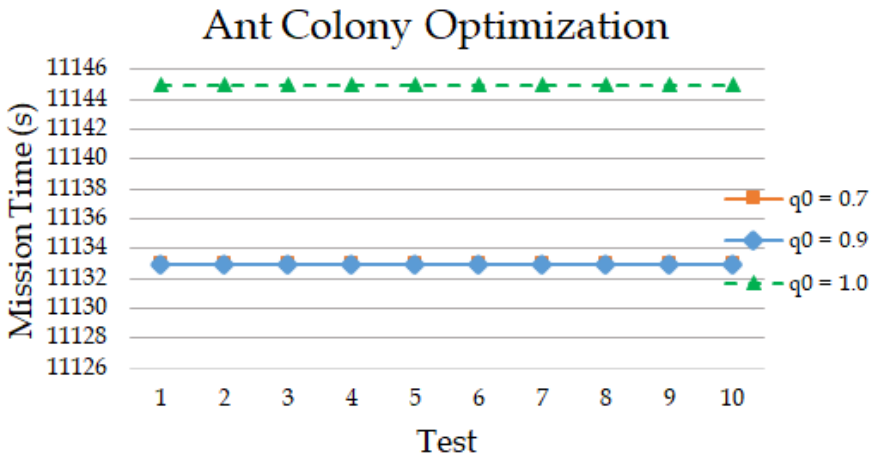


Figure 5.35: Total Mission Time for each test varying q0 threshold.

The parameters thus outlined are those in the Table 5.9, tested also for other different maps; however, if the number of cells increases significantly (number of total regions > 500) it would be advisable to perform further tests to reset the parameters, as the convergence in that case would differ greatly. In any case, the solution could be to keep a fairly low number of regions per UAV, therefore by increasing the total area, i.e. the number of regions, it is possible to opt for an insertion of drones rather than a reset of the entire method.

	Number of Ants	**Evaporation Ratio**	**α parameter**	**β parameter**	**q0 threshold**
Value	5	0.2	1.0	2.0	0.9

Table 5.9: Final Parameters for the Ant Colony Optimization

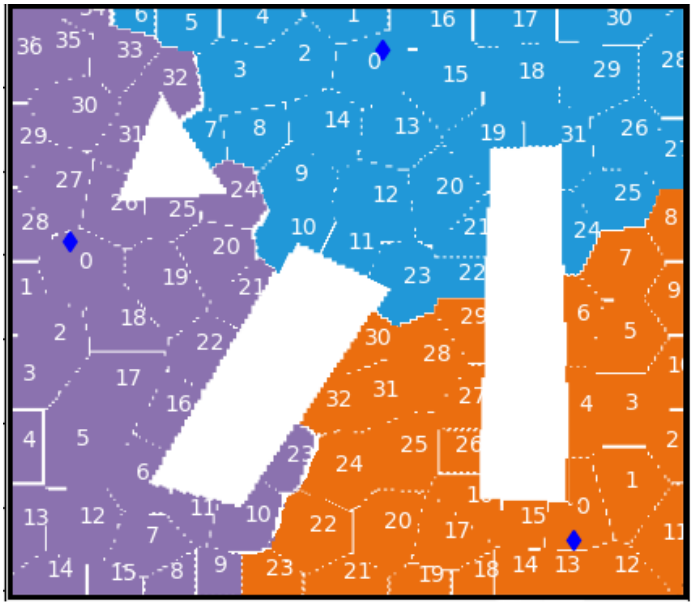


Figure 5.36: Best order using Ant Colony for each UAV for selected map.

After selecting the final parameters, further generic maps were generated, also using more drones, generating random obstacles within the operational area. All the following Figures are examples also with drones with different performances.

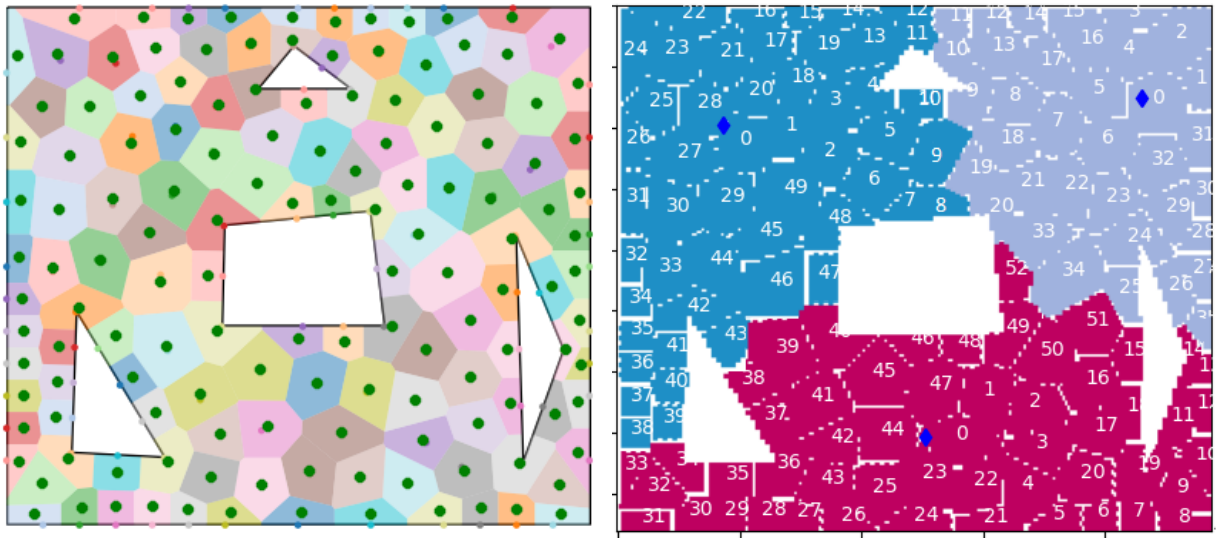


Figure 5.37: Voronoi diagram and region coverage for random map.

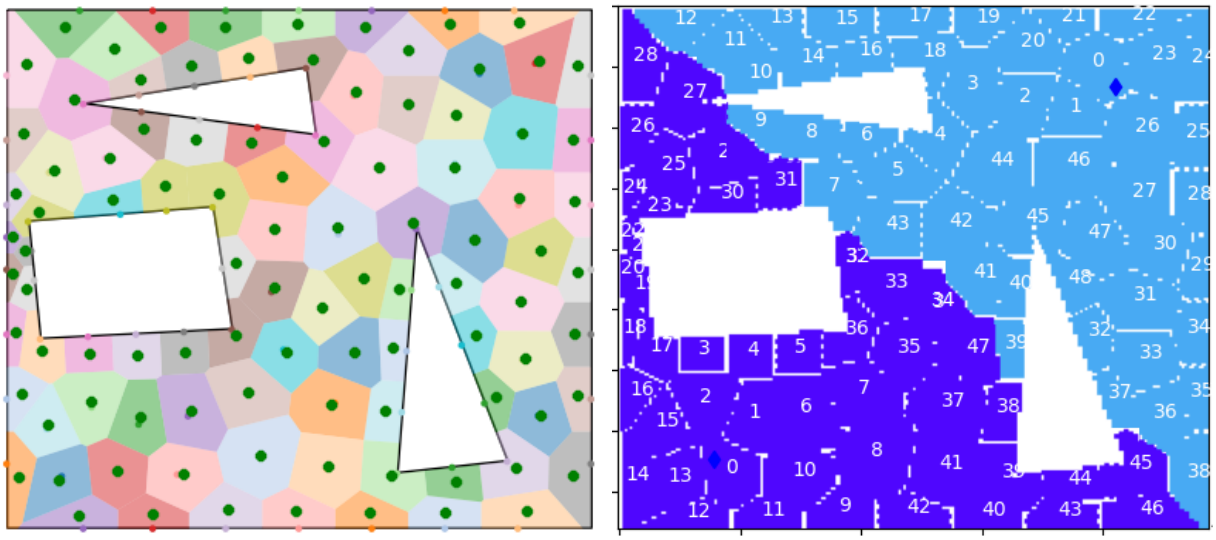


Figure 5.38: Voronoi diagram and region coverage for random map.

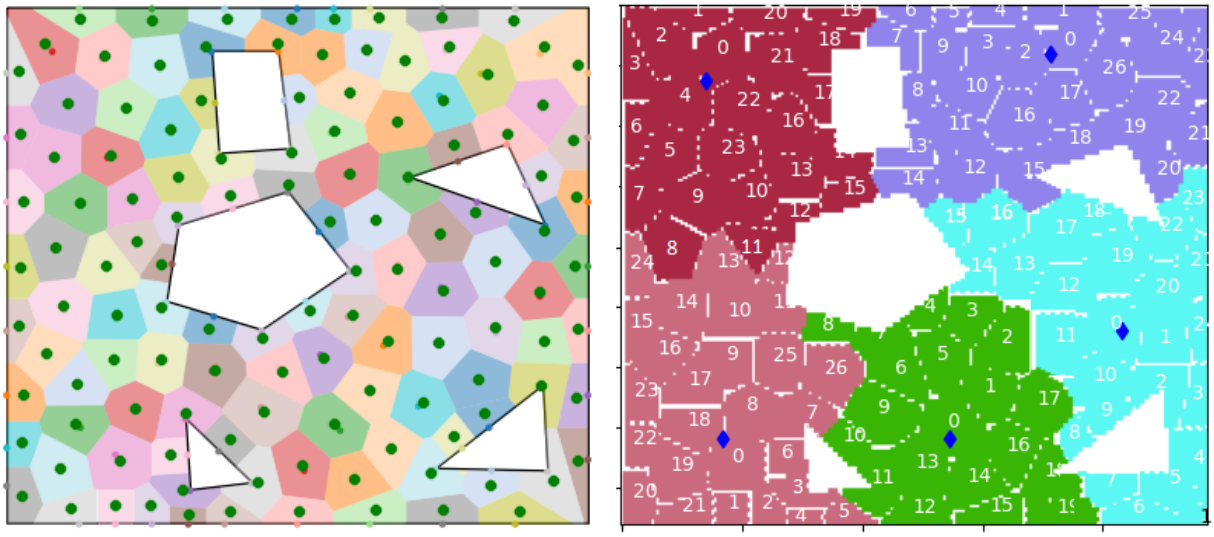


Figure 5.39: Voronoi diagram and region coverage for random map.

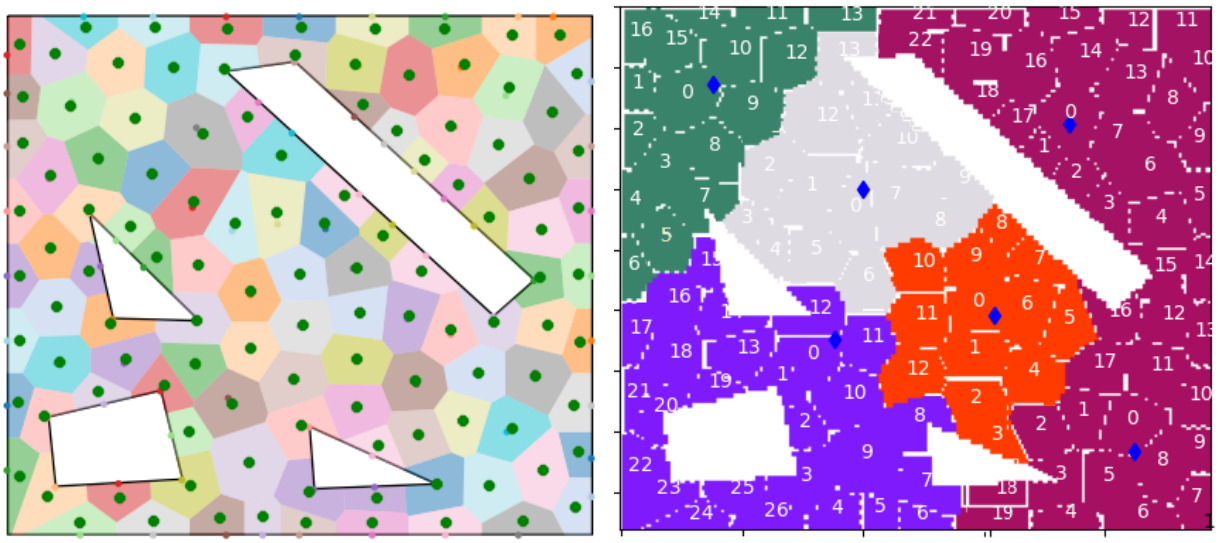


Figure 5.40: Voronoi diagram and region coverage for random map.

Chapter 6

Conclusions and Future Developments

This thesis work allowed to investigate the various methods related to Coverage Path Planning (CPP) through the use of multi-robot systems. The strategy adopted was composed of two steps, an initial one relating to the subdivision of the operational area into several regions, and a subsequent step in which the regions generated are allocated for each UAV and ordered. In the former, Boustrophedon Cell Decomposition was used firstly for the subdivision of the operational area, later, it was adopted the Voronoi diagram to obtain improvements in the performance of the whole multi-UAV system. Subsequently, after having generated some sub-regions, the Ant Colony algorithm was adopted for the allocation and sorting of the cells for each UAV, with the final tuning of the parameters for the optimization of the entire algorithm.

Overall, the coverage planning algorithm turns out to be very valid as the number of drones and the number of obstacles present within the operational area increase. The choice to use the Voronoi diagram, with the aid of the Delaunay triangulation, has led to very satisfactory results even compared to other methods present at the state of the art. As regards the assignment and ordering through the algorithm based on the Ant Colony, the effectiveness of the method is observed to have the convergence of the most immediate solution with respect to alternative algorithms, and subsequent developments can be carried out regarding the collision avoidance and communication between drones in an online flight control algorithm, also including the altitude parameter for the study of the subdivision of the area and the motion of the drones.

Finally, as regards the entire algorithm, future works may include a more in-depth study of the method of visiting the individual regions, thus combining the coverage planning algorithm with one for the visit of a single area, thus reaching an optimal solution for both subdivision and motion within the entire operational area and improving of the results in the previously reported critical issues. Further studies may concern the interaction of the algorithm with a realistic simulation performed using Robot Operating System (ROS), a popular framework consisting of libraries and applications for robotic programming.

List of Figures

List of Tables

Bibliography

[1] Nedjati A. et al. "Complete coverage path planning for a multi-UAV response system in post-earthquake assessment." In: *Robotics* 5 (2016), pp. 4–26. DOI: https://doi.org/10.3390/robotics5040026.

[2] Vemprala S. and Saripalli S. "Vision based collaborative path planning for micro aerial vehicles." In: *IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD* (2018), pp. 1–7. DOI: https://doi.org/10.1109/ICRA.2018.8462910.

[3] Cabreira T.M, Brisolara L.B., and Ferreira P. R. Jr. "Survey on Coverage Path Planning with Unmanned Aerial Vehicles." In: *Drones* 3 (2019). DOI: 10.3390/drones3010004.

[4] Choset H. "Coverage for robotics—A survey of recent results." In: *Ann. Math. Artif. Intell.* 31 (2001), pp. 113–126. DOI: https://doi.org/10.1023/A:1016639210559.

[5] Di Franco C. and Buttazzo G.C. "Coverage path planning for UAVs photogrammetry with energy and resolution constraints." In: *J. Intell. Robot. Systems* 83 (2016), pp. 1–18. DOI: https://doi.org/10.1007/s10846-016-0348-x.

[6] Cabreira T.M. et al. "Energy-Aware Spiral Coverage Path Planning for UAV Photogrammetric Applications." In: *IEEE Robot. Autom. Lett* 3 (2018), pp. 3662–3668. DOI: 10.1109/LRA.2018.2854967.

[7] Galceran E. and Carreras. "A survey on coverage path planning for robotics." In: *Robotics and Autonomous Systems* 61 (2013), pp. 1258–1276. DOI: https://doi.org/10.1016/j.robot.2013.09.004.

[8] Oksanen T. and Visala A. "Coverage path planning algorithms for agricultural field machines." In: *Journal of Field Robotics* 26 (2009), pp. 651–668. DOI: https://doi.org/10.1002/rob.20300.

[9] Torres M. et al. "Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction." In: *Expert Syst. Appl.* 55 (2016), pp. 441–451. DOI: https://doi.org/10.1016/j.eswa.2016.02.007.

[10] John Milnor. *Morse Theory.* Princeton University Press, 1963.

[11] Barrientos A. et al. "Aerial remote sensing in agriculture: A practical approach to area." In: *J. Field Robot* 2 (2011), pp. 667–689. DOI: 10.1002/rob.20403.

[12] Kantaros Y., Thanou M., and Tzes A. "Distributed coverage control for concave areas by a heterogeneous robot-swarm with visibility sensing constraints." In: *Automatica* 53 (2015), pp. 195–207. DOI: https://doi.org/10.1016/j.automatica.2014.12.034.

[13] Ranjbar-Sahraei B., Weiss G., and Nakisaei A. "A Multi-Robot Coverage Approach based on Stigmergic Communication." In: ().

[14] Manjanna S et al. "Heterogeneous multirobot system for exploration and strategic water sampling." In: *IEEE international conference on robotics and automation (ICRA'18)* (2018), pp. 4873–4880. DOI: 10.1109/ICRA.2018.8460759.

[15] Sánchez P, Casado R, and Bermúdez A. "Real-Time Collision-Free Navigation of Multiple UAVs Based on Bounding Boxes." In: *electronics* 1632 (2020). DOI: 10.3390/electronics9101632.

[16] Almadhoun R et al. "A survey on multi-robot coverage path planning for model reconstruction and mapping." In: *SN Applied Sciences* (2019). DOI: https://doi.org/10.1007/s42452-019-0872-y.

[17] Sánchez P, Casado R, and Bermúdez A. "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms." In: *Distributed Autonomous Robotic Systems* 6 (2007), pp. 221–230. DOI: 10.1007/978-4-431-35873-2_22.

[18] Vincent P. and Rubin I. "A Framework and Analysis for Cooperative Search Using UAV Swarms." In: *Proceedings of the A Framework and Analysis for Cooperative Search Using UAV Swarms,* (2004), pp. 79–86. DOI: https://doi.org/10.1145/967900.967919.

[19] Kaminka G. and Agmon N.and Hazon N. "Constructing spanning trees for efficient multi-robot coverage." In: *Proceedings 2006 IEEE International Conference* (2006), pp. 1698–1703. DOI: 10.1109/ROBOT.2006.1641951.

[20] Fazli P. et al. "Complete and robust cooperative robot area coverage with limited range." In: *IEEE/RSJ 2010 international conference on intelligent robots and systems* (2010), pp. 5577–5582. DOI: https://doi.org/10.1109/IROS.2010.5651321.

[21] Kapoutsis AC., Chatzichristofs SA., and Kosmatopoulos EB. "DARP: divide areas algorithm for optimal multi-robot coverage path planning." In: *J Intell Robot Syst Theory Appl* 86 (3–4) (2017), pp. 663–680. DOI: https://doi.org/10.1007/s10846-016-0461-x.

[22] Godio S. et al. "A Bioinspired Neural Network-Based Approach for Cooperative Coverage Planning of UAVs." In: *information* 12, 51 (2021). DOI: https://doi.org/10.3390/info12020051.

[23] Chen J. et al. "Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system." In: *Elsevier B.V* 69 (2022). DOI: https://doi.org/10.1016/j.swevo.2021.101005.

[24] Dorigo M. and Gambardella LM. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". In: *IEEE Transactions on Evolutionary Computation* 1 (1997), pp. 53–67.

[25] Daniel J. Rosenkrantz, Richard Edwin Stearns, and Philip M. Lewis. "An Analysis of Several Heuristics for the Traveling Salesman Problem". In: *SIAM J. Comput.* 6 (1977), pp. 563–581.