Guaranteeing Quality of Service in Persistent Area Monitoring Using Multi-UAV

Miguel Carvalho, Luis Ramos Pinto, Rodrigo Ventura

***Abstract*— Nowadays, Unmanned Aerial Vehicles (UAVs) are popular agents in many applications. Using these drones allows some tasks to be completed more safely and consistently. Such applications include area monitoring, reconnaissance and item transportation. It is an objective of area monitoring that the information on a given area is as fresh as possible. When deploying a fleet of UAVs in the Area of Interest (AoI), the complexity of this area can prove to be a challenge to the mission completion. Instead of having complete movement freedom, the UAVs might be restricted in certain regions. The priority of sections of these AoIs can also differ in the same application. This dissertation addresses the periodic monitoring of an AoI. The AoIs can be of different levels of complexity, requiring different amounts of UAVs. The higher the AoI's complexity, the more challenging is to find a feasible solution for all time and space constraints. Multiple paths can are generated to be an initial solution for each problem. The initial conditions are compared in multiple scenarios to achieve the best results. An algorithm is proposed to control the paths of every UAV deployed in the AoI. The resulting UAV paths guarantee that all time constraints of the AoI are respected. It's shown that in more complex areas, the need for more than one UAV is more evident. Besides, at a certain point, when increasing the number of UAVs, the improvement in the results stops being relevant.**

Keywords - Deadlines, Multi-UAV, Path Allocation, Persistent Monitoring, TSP

I. Introduction

Nowadays, there are multiple types of drones for every environment: aquatic, ground or aerial. The aerial drones are also known as Unmanned Aerial Vehicles (UAVs). There are multiple classes of UAVs, for example, Fixed-Wing, Monorotors and Multirotors. In this work, the focus is on using autonomous multirotor UAVs.

UAVs can be used in multiple applications. If an area needs to be monitored regularly, it is of most interest that UAVs are used to fly over a said area, especially if the access is difficult or of high risk for humans. Currently these vehicles are used in multiple areas of application, for example, military reconnaissance, merchandise transportation and aerial surveys. For each application, UAVs are attached with sensors to perform the required tasks.

There are various applications of UAVs in Area Coverage. In Agriculture, drones can monitor large portions of terrain, measuring soil characteristics or harvesting. In Vigilance of high security areas, using multiple drones leads to fewer blind spots. The danger in monitoring areas contaminated with high levels of radioactivity can be reduced when using drones.

When deploying a group of UAVs in an area, there might be some factors that difficult the drones' flight, like obstacles. The complexity of an area can highly influence the UAVs' movement and thus compromise the mission.

This work is focused on the topology of the area to be monitored. The topology is related with the complexity of said area.

Area Complexity can be related to time and space. For example, having some regions declared as restricted (or having obstacles) will force the drones to find a way around these zones. On the other hand, regions that require to be monitored more frequently will increase the time complexity of the problem.

The goal of this work is to study the use of a group of UAVs to monitor an area of interest. This area is meant to be monitored periodically, bearing in mind all its time and space constraints.

This paper serves as an extended summary of the work developed in the corresponding Masters Dissertation. It is organized as follows: First, in Part II, the work that is expanded by the dissertation is described as well as the state of the art in the field. In Part III, it is described the problem under study and what kind of solution is being searched. The theoretical approach to the problem is presented in Part IV and in Part V it is explained how the solutions are implemented. At the end, in Part VI, some results are presented.

miguel.a.carvalho@tecnico.ulisboa.pt
luis.pinto@mov.ai
rodrigo.ventura@isr.tecnico.ulisboa.pt

II. Related Work

The work developed in the dissertation expands the one in [1], where solutions for having a group of UAVs monitoring an area of interest are studied. The goal was to transmit the data collected by the drones to a base station, having its information as up to date as possible. To ensure the information is quickly updated, multiple UAVs are used to cover the area.

The authors present an iterative solution to generate one shortest route to cover all the AoI. There are few variations when the length (in cells) of the area is even or odd. This route is based on an Hamiltonian Cycle, [2].

An Hamiltonian Cycle is a route through a graph that transverses all the nodes and returns to the starting node.

The authors of [1] implemented an iterative solution for the Hamiltonian Path. Starting from a corner of the AoI, go around its edge and then zig-zag the interior until the drone reaches the initial position.

One of the conclusions from [1] is that using a group of drones well spread along a singular path is sufficient to guarantee that the information on the area is as up to date as possible. In [1], the considered areas were empty, meaning the movement of the drones was not obstructed by obstacles.

The objectives of this dissertation is to investigate if the conclusions in [1] hold, in terms of having only one path for the drones, when facing more complex areas of interest. It tries to find a strategy that satisfies the monitoring specifications.

In multiple works, different strategies can be compared to the ones that are used in this dissertation.

In [3], multiple Hamiltonian paths are used to deploy a group of Autonomous Guided Vehicles (AGVs) in a factory. The paths are improved with a Genetic Algorithm.

[4] uses a UAV to visit a number of objects persistently. Each object is associated a different deadline representing the the frequency at which new information appears in it. This frequency is related with the priority of the object. The work considers multiple heuristics involving Earliest Deadline First (EDF), Slacks and K-steps (searching steps ahead).

In [5], multiple strategies are presented to use the TSP for multiple agents.

In [6], the Multiple TSP (mTSP) is solved using an heuristic based in Ant Colony Optimization (ACO). It focus on spreading the agents through the area, avoiding crossing paths as much as possible.

The work in [7] uses an iterative algorithm to create a path for a group of UAVs. The paths consist in following a direction until there is the need to change it. These paths are not cyclic. It is guaranteed that all paths don't overlap.

In [8], a region is divided into sections based on the starting positions of a group of robots. One of the goals is to create even sections. Introducing obstacles in the area might cause some problems to be unsolvable by the presented method.

[9] tackles an Energy problem when monitoring an area. It uses a fleet of small UAVs, travelling at different velocities, to maximize the number of up-to-date information given a limited number of batteries. Multiple routes are constructed using insertion heuristics. The points to be monitored are associated with priorities. Picture points are visited according to their priority.

The work in [10] uses multiple drones to visit points in an area susceptive to external events. These events are handled with different priorities. The tasks related to the events can be shared by more than one agent.

In [11], a group of robots is used to visit points with different frequencies in an area. The robots negotiate their paths upon completion to cooperate in covering the whole area.

[12] uses mobile sensors in a Wireless Sensor Network (WSN). The Area Coverage is done under time constraints. Each Mobile sensor has a limited time to cover as much area as possible before having to return to a base station.

III. Problem Description

This dissertation focus is on the complexity of an Area that needs to be monitored periodically.

To accomplish Area Coverage, multiple UAVs are deployed in the area of interest, each with a well defined path. Periodicity is achieved by repeating the drones' paths.

The goal of this dissertation is to generate as many paths as there are drones. Every must guarantee that every portion of the Area is visited respecting the imposed time constraints.

The solution is found in a centralized manner and provided to the UAVs before they are deployed in the field. It is calculated based on information on the dimensions and time constraints of the area, number of UAVs and their velocity. All distances in the area where the monitoring is done are known.

The UAVs' paths are a set of 3D points comprising the 2D coordinates describing the location where the drones sense the data and the time instant when that sensing must occur.

A. The Area of Interest

The Area of Interest (AoI) is modelled as a two dimensional and quadrangular grid with dimensions $W \times W$, where W is the side of the AoI.

Each element of the grid is defined as a cell. In the AoI there are $M \times M$ cells of dimensions $L \times L$. Then, M can be seen as the integer division of the side of the AoI in Cells. These dimensions for the cells are related to the Area of Effect (AoE) of the UAVs in use. These are represented in Figure 1.

Each Cell is identified by a number and has its coordinates. The coordinates represent the location of the Cell in the AoI and correspond to that Cell's centre.

A path is an ordered set of Cells representing a sensing order.

The UAVs must hover above every Cell in their path and gather any information needed (sensing the cell). When a UAV senses a Cell, it is hovering above the centre of said Cell. The time a drone requires to sense a Cell is fixed.

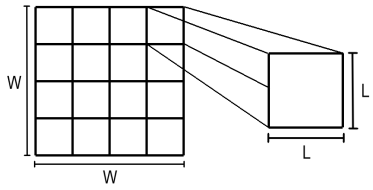

Fig. 1. Dimensions of the grid

One way to increase complexity in the AoI is the introduction of specific points that are not required to be sensed. There can also be zones where the correct acquisition of information is impossible. Cells with these characteristics can either be viewed as restricted areas or obstacles.

An example of an application of this representation is presented in Figure 2. It's a plantation field where only some kinds of plants are pretended to be monitored. The flight of the UAVs is restricted in the regions marked in red.

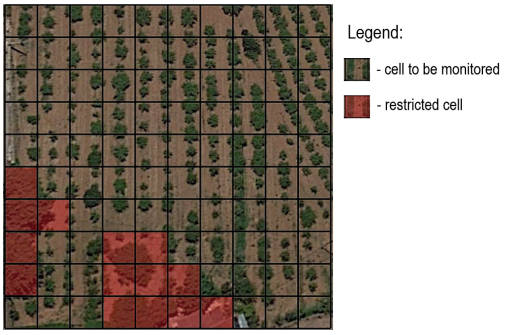

Fig. 2. Representation of the AoI as a grid with restricted cells

Setting a maximum frequency with which the Cells can be visited also increases the AoI's complexity.

This time constraint is defined as the Deadline of the Cell.

During each mission, every Cell is visited at least one time. The order in which Cells are monitored, i.e the path of each UAV, is influenced by their Deadline.

The Deadline is the maximum amount of time that a Cell can wait to be visited by one of the drones. Respecting all deadlines is the focal point of this work. A Cell cannot be without being visited for more than its Deadline.

B. The UAVs' Movement

The considered UAVs in this work are quadrotors.

In each mission, there are N identical UAVs ready to be used. It is assumed that the drones fly at the same height and travel at a constant velocity. The velocity of the drones is modelled as the time a drone takes to go from one Cell to an adjacent Cell. A drone is hovering over a Cell when it senses it.

It is assumed that, in order to avoid collisions, if two drones are on the same Cell, it is assumed that they pass the Cell at slightly different heights.

Delays in the visit times of the Cells are not considered.

IV. Theoretical Approach

The AoI as a quadrangular grid is converted to as an undirected graph, as in Figure 3. The nodes in the graph represent the centre of the Cell and the edges represent a path between two adjacent Cells.

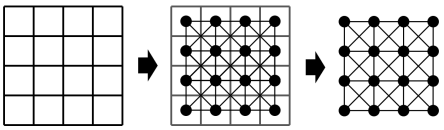

Fig. 3. Graph representation of the AoI

The distance between adjacent cells is then a approximation of Euclidean distances. It is calculated using the cells' geometric centres. The Euclidean distance between two points is given by Equation 1.

$$d(p,q) = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2} \tag{1}$$

It is considered that two adjacent cells in the same row or column are 1 unit of distance (u.d.) apart. As for the diagonals, the value is rounded to the tenths. In this case, diagonally adjacent cells will be 1.4 u.d. apart.

Restricted Cells in the AoI will be represented as singletons so they are not connected to other Cells.

Every Cell to be sensed by the drones must be reachable. For that, all these Cells must form one connected component of the AoI's graph.

A. Initial Solution

A path in a graph is a sequence of vertices connected with edges. The shortest path is such that the sum of the weights of the edges between two vertices is minimized. The edges can have weights representing some characteristic of the problem. In the case in study, the weights are the distance between adjacent cells in the AoI.

A well known problem in graph theory is the Travelling Salesman Problem (TSP). The TSP views every node as a city and every edge as the road that connects each city (known distance). Its goal is to find the shortest path for a salesman to take such that they visit every city only once.

The path obtained from the TSP seems to be a good starting point for the problem in this dissertation. However, in the AoI, there might be cells to be monitored more frequently, which requires that what the TSP sees as cities would have to be visited more than once. After acquiring the path with the TSP, some adaptations are needed regarding these Cells.

Many algorithms calculate distances in a graph. To calculate the distances and thus compute the solution for the TSP, it's used the Dijkstra Algorithm.

The Dijkstra Algorithm finds the shortest path between a node and every other node. This algorithm has complexity $\mathcal{O}(n^2)$, where n is the number of nodes in the graph. When running the algorithm for every node, the complexity becomes $\mathcal{O}(n \times n^2)$, i.e $\mathcal{O}(n^3)$.

There is other path that is considered, although with some modifications. This path is identified as Baseline and is used in [1]. It can be viewed in Figure 4.

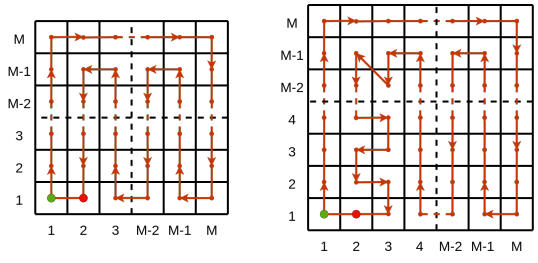

Fig. 4. Baseline Path for even (left) and odd (right) values for M

When there are restricted cells in the AoI, the Baseline path should have the same framework but must ignore these cells. This only affects the path taken by the UAVs and not the order in which the Cells are visited.

Special Cells need to be visited more often. With regards to each Cell's priority, the number of additional visits they require is defined by the Class to which they belong. For example, if a Cell belongs to Class 2, then it must be visited an additional time while most Cells are only visited once.

Whichever the initial path chosen for the drones, it is rearranged to assign a different path for each drone. The rearranged paths allow for the drones to be spread evenly throughout the AoI.

B. Iterative Method

There are two goals in the Problem, given an initial path for each drone. The first is to check if the initial paths guarantee the Deadline specifications. The second is to modify the paths when the first goal is not met.

The Iterative Method modifies each Drone's path one step at a time in order to achieve these goals.

The modifications in the paths are made when a Cell would be visited after the Deadline expired. The goal is to make the drone that detects this fact to visit this Cell earlier in its path.

For that, when it is detected that a Deadline was not respected, the drone will visit that Cell before the previous Cell it visited.

Every time a swap is made, the time of the mission is regressed.

After a certain number of visits to every Cell, the Iterative Algorithm is stopped.

V. Implementation

This Problem being studied in this work consists in finding one path for each agent deployed in the AoI. It was developed a tool, in MATLAB, that represents the AoI and monitors the Deadlines of each Cell throughout a mission. The tool's goal is to provide insight about two aspects of the problem. One is the average time a Cell stays without being visited. The other is the path for each UAV that ensures every Deadline is respected.

The tool has knowledge about the number of the UAVs (N) and the topology of the AoI. The topology of the AoI comprises information about the number and location of the Cells. It is provided the number and location of possible restricted Cells and the deadlines for the cells to be monitored. It is possible to compute solution for multiple AoIs of different sizes and multiple groups of UAVs also with different sizes. The user of the tool must input some information about the mission. With that information, it is possible to find the paths. Two operation modes were developed in the tool. The first attributes a path to each drone and simulates a mission where the drones would follow that same path. The second mode also attributes a path to each drone, but in this case the Deadlines are monitored. The user proceeds to indicate the existence of Restricted Cells or Cells with a Deadline smaller than the default Deadline (Special Cells). These Cells can also be generated randomly, being the maximum amount of Cells or a certain type set to 30% of the number of Cells in the AoI. Lastly, the user must choose the type of initial path to be considered for the drones. There are only two options. One is to use the path framework that was used in [1]. The other is to solve the TSP for each considered AoI. The tool's outputs are the paths for every drone in every group of drones, a graphic representation of the movement of the drones during the beginning of the mission and histograms to analyse the time between visits of Cells in the same Class. The pseudo-code for the tool's execution is presented in Algorithm 1.

```
Define AoIs' sizes;
for Every AoI size do
    Build graph;
    Identify Restricted Cells;
    Identify Special Cells;
    Choose Deadlines;
end
Define UAV groups;
for Every AoI size do
    for Every Group of UAVs do
        Generate Initial Path;
        if > 1 UAV then
            Segment Initial Path to spread UAVs in the AoI;
        end
        Execute Mission;
        if Feasible then
            Show Results;
        end
    end
end
```

Algorithm 1: Pseudo-code for the MATLAB tool Execution

A. Problem Representation

The number of Cells in the AoI is defined by the user with the value M. An undirected graph representing the AoI is created with M^2 vertices. Upon selecting the Restricted Cells, the edges that connect the respective edges are deleted from the graph, maintaining the vertices. When defining the Cells with a different Deadline, the user identifies them as they did for the obstacles. A different Deadline can be defined for each Special Cell. All relevant data about the AoIs, the UAVs and the Cells is saved in the respective data structures.

Every Cell is associated with a different timer, which controls the Cell's Deadline (D). Every time a cell is visited by either one of the drones, the timer is reset to zero. All Cells, apart from Restricted Cells, are grouped in Classes from 0 to 5. The Cell's Class is related to the classification of the deadlines based on their difference from the Default. Classes are seen as a measure of priority. The higher the Class, the higher the priority. Class 0 corresponds to cells with a Deadline higher than the Default Deadline. The Default Deadline is determined with Expression 2.

$$\text{DefaultDeadline} = M^2(t_{\text{sensing}} + t_{\text{flying}}) \tag{2}$$

The ranges for the other classes are defined by Equations (3) and (4):

$$\text{LowerBound} = \text{DefaultDeadline} - \text{Class}\frac{\text{DefaultDeadline}}{5} + 1 \tag{3}$$

$$\text{UpperBound} = \text{DefaultDeadline} - (\text{Class} - 1)\frac{\text{DefaultDeadline}}{5} \tag{4}$$

A drone's path is defined before the mission starts. An initial solution is generated using the TSP or the Baseline method. When there are more than 1 UAVs being used, the generated path is segmented and rearranged to distribute the UAVs in the AoI. An example of the path assignment is shown in Figure 5. In this example, the Baseline path is generated for an AoI with 9 Cells and paths for 2 drones are created.

Each drone has two modes of operation: "Flying" and "Sensing". In the "Flying" mode, the UAV moves from one Cell to the other until it reaches the next Cell to sensor. The drones will also be in this mode at the beginning and end of each mission. When in "Sensing" mode, the UAVs are hovering over a Cell that is being sensed. Only one drone can monitor the same cell at once. The array State saves the time instants when a drone changes its mode and to which mode it changes to.

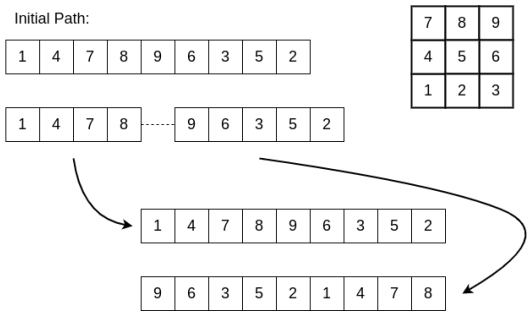

Fig. 5. Distributing 2 UAVs in the AoI

B. Initial Solution Path

A path is an ordered set of Cells that a drone needs to sense. At the start, all drones are in the Flying State and move to the next cell that they must sense. At every time instant, all drones are monitored by the tool. Monitoring a drone is knowing the drone's position, its State and having an estimation of when this state will change.

Since all distances between all Cells are known, it is possible to determine the time a drone takes to fly from one Cell to the other. The velocity of the drones is constant and is adjusted with the constant t_{flying}. This constant represents the amount of units of time required to transverse one unit of distance in the AoI. Equation 5 expresses the time instant when a drone gets to a Cell B, where t is the current time instant and d is the distance between Cell A and Cell B.

$$t_B = t + t_{\text{flying}} * d(A, B) \tag{5}$$

If a drone in the Flying State, its position is determined to the cell level. When the estimated cell is reached, the drone's State changes.

If, on the other hand, a drone is in the Sensing State, the algorithm checks if the current instant is the estimated instant for the sensing of the Cell where the drone is to end. A drone starts sensing a cell in the time instant after its state changes to sense. The sensing duration is set by the constant t_{sensing}. This constant represents the amount of units of time required by the drones to sense a Cell.

When the drone completes that cell's visit, information on the fact is updated and registered. After this, the next cell to where the drone must move is determined. While a drone is still visiting the Cell, the Cell's Timer will freeze, maintaining the value it had when the drone arrived.

C. TSP with MATLAB

In a MATLAB environment, a solver-based solution [1] was selected to compute the TSP path. The solver uses Binary Integer Programming (BIP). The solution is a shortest tour that goes through every node in a graph. It is found by creating smaller subtours that are then merged to form the wanted tour. When there is only one subtour, it is guaranteed that a drone can have a shortest path through the entirety of the AoI. Assuming that all drones are deployed from the same point, it is sufficient to define one path at this point. Having the cell where the drones are deployed, a Depth First Search (DFS) Algorithm is run from the node corresponding to that cell.

[1] https://www.mathworks.com/help/optim/ug/travelling-salesman-problem.html

The DFS is applied to the final tour, a graph with every node having only two edges. A node corresponding to the starting position is taken as the root. This algorithm gathers the cells in order to what will be considered the initial path. Since this TSP gives a graph where a cycle can be described, when the DFS reaches the end, it is near the root node again.

At the end of the process, there is a path to travel in the AoI. This path accounts for every Cell and minimizes the time spent moving by the drones.

D. Solving the TSP in larger AoIs

The TSP is only ran once for each mission but, given the complexity of the problem, it's a lingering operation when the size of the AoI increases. The increase in computation time starts to get noticed when the size of the AoI exceeds $M = 10$. To limit the time spent generating the TSP for the whole AoI, it is split into four sections. A TSP adequate to each section is then calculated. Since all sections will be stitched together at the end, to form a continuous path, each TSP solution starts and ends as close as possible to the center of the AoI. In Figure 6 an example of a AoI split can be viewed, as well as the start and finish of each TSP tour.

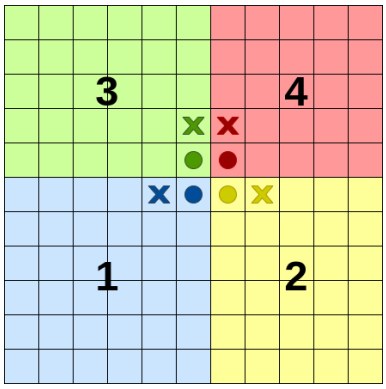

Fig. 6. Area division for TSP with M = 11
(o - start of route; × - end of route)

Finally, all the paths are stitched together. This is done starting from section one. Then, the path corresponding to the next section is attached at the end of the former section's path, following a clockwise order.

E. Iterative Algorithm

The Iterative Method is implemented using what was called the Iterative Algorithm, shown in Algorithm 2.

All drones start the mission in the same position. They must move to the first Cell in their path. That is the first Cell to be sensed. The initial State is the Flying State.

As the time to sense (t_{sensing}) is considered to be constant, it can be known when the Cell's sensing will be finished. The estimation of the time instant when the Cell is sensed (t_{fSen}) is done using Equation (6). In (6), t_{arrC} is the estimated time when a Drone gets to the Cell and begins sensing. While the drone is not done sensing a Cell, the Cell's Timer remains constant at the value it had when the drone arrived.

$$t_{\text{fSen}} = t_{\text{arrC}} + t_{\text{sensing}} \tag{6}$$

When a Cell finishes being sensed, the Drone enters the Flying State. When the drone enters the new state, it needs to determine its next Cell to sense. Determining the next Cell involves knowing

```
Initialization;
while In Mission do
    for Every Drone do
        if Sensing a Cell then
            if Finishes Sensing then
                Update Visit;
                Determine Next Cell to Sense;
                Change to Flying State;
            else
                if Cell Occupied then
                    Determine Next Cell to Sense;
                    Change to Flying State;
                else
                    Change to Sensing State;
                end
            end
        end
    end
    Verify Stop Condition;
end
```

Algorithm 2: Pseudo-code for the Iterative Algorithm

where every other drone is. A Drone will only travel to the next Cell in its path if there is no other drone that either is currently sensing that Cell or will arrive to it before the current drone does.

Periodicity in following a drone's path is arranged by duplicating it when the drone senses the last Cell. Doing this provides a larger margin to reorder the path when a Deadline is not respected.

When a Drone is in the Flying State, the current time instant is compared with the estimated time of arrival to its next Cell. When it arrives to the Cell, the Cell's Timer is checked. If the Timer exceeds the Cell's Deadline, it means that the Cell has been without being sensed for more time than it should. In this case, the Drone that finds this out will try to reach this Cell earlier.

The Iterative Algorithm handles the Deadlines by swapping the order of visitation of Cells in a Drone's Path. A swap in a drone's path does not assure that moving to this Cell earlier prevents other cells' Deadlines to not be respected. When a swap in a path is made, mission time will regress to the instant the drone finishes sensing the Cell before the previous. When a repeated swap is detected, the Problem is declared Unfeasible.

If every Cell is visited the minimum required times, then the sum of all counters is the Number of Effective Cells, in (7). It sums the total number of Cells in the AoI with the extra visits of Special Cells and subtracts the Number of Restricted Cells (N_{Obs}). For this calculation, the Special Cells are considered to require only one more visit so the extra is only the Number of Special Cells (N_{Spec}). The chosen Stop condition for a solvable problem is having the sum of these counters to be ten times the Number of Effective Cells.

$$N_{\text{Eff}} = M^2 - N_{\text{Obs}} + N_{\text{Spec}} \tag{7}$$

VI. Results

In this summary, some results are gathered that demonstrate the performance of the Implemented Methods. In the dissertation, the considered initial paths are compared. It is presented an analysis on the computation time required to generate an initial path and the time a singular drone takes to complete it. To analyse the performance of the Iterative Method, multiple AoIs are tested. These AoIs vary in size and can be of three types: empty (an AoI with no Restricted or Special Cells), AoIs with Special Cells, AoIs with both Restricted

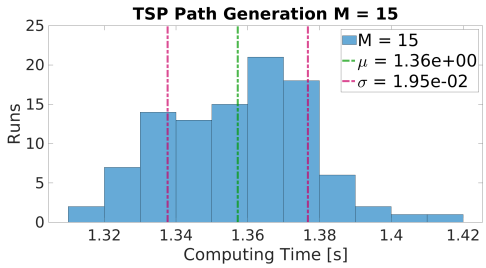

Fig. 7. Histograms of Running Times for generating the TSP path in an AoI with no Restricted or Special Cells

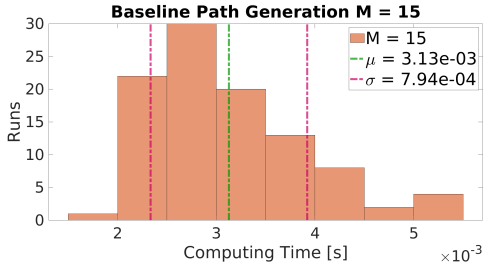

Fig. 8. Histograms of Running Times for generating the Baseline path in an AoI with no Restricted or Special Cells

and Special Cells. For the tests in the Empty AoIs, the used values for both t_{sensing} and t_{flying} are 20 and 30 u.t., respectively. In these tests, 1 unit of time is equivalent to 0.1 seconds. After this, some alternatives are studied to handle unfeasible problems.

A. Comparison of Initial Paths

To decide which path is better, one can start by comparing the time it takes for both to be computed, given the same AoI. The AoI has no restricted cells so the paths include every cell in the AoIs. Results were generated for different sizes of AoI setting $M = \{5, 10, 15, 20\}$.

1) Running Times: For each of the AoIs, each path was calculated 300 times. From each set, the outliers were removed using the Mean Absolute Deviation (MAD) method. From the resulting set, were selected 100 runs, randomly. As an example, the results obtained for an empty large AoI ($M = 15$) are in Figures 7 and 8. As expected, the time to compute the paths increases with the resolution (value of M) of the AoI. It is possible to prove that the Method to generate the TSP for larger AoIs highly improves the time required by the task. It is clear that the Baseline path is generated much faster than the TSP path.

2) Time to Complete Path: A drone completes a route every time it visits every cell in its path at least once. Being the initial paths different, since the Baseline Path doesn't take into account the presence of Restricted Cell, it is interesting to study the performance of the drones completing a route in AoIs with this type of Cells. For the tests, t_{flying} and t_{sensing} are constant. The paths are compared in multiple AoIs with different sizes and amounts of Restricted Cells. The time to complete both paths in different large AoIs with 20% Restricted Cells is presented in 9.

With the Baseline path, UAVs have to constantly go around the Restricted Cells in order to reach the next Cell. This doesn't happen with the TSP and the time needed is very similar for every AoI.

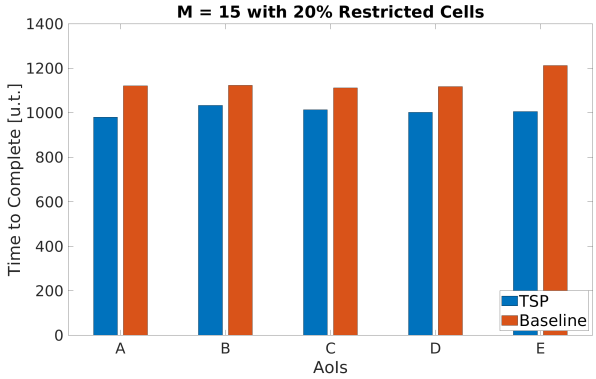

Fig. 9. Time to complete a route in large AoIs with 20% Restricted Cells for both paths

From what has been seen, choosing the TSP over the Baseline is a viable solution. The larger the AoI the better the results are in terms of Time to Complete a Route. Adding the fact that the drones should follow a path periodically, the difference in times will become more evident. The time needed to compute the TSP in the AoIs once, apart from some special cases, is recovered in time saved when moving the UAVs.

B. Fleet Performance in Empty AoIs

It is generated a TSP path as the initial path. The path is assigned to different groups of UAVs. The groups contain from 1 to 5 UAVs. The paths and visit frequency of every Cell is compared with the increase of the number of UAVs. For the Deadline of Class 1 Cells, the Default Deadline is used. In large UAVs, this Deadline is 1125 seconds. It is impossible to use one UAV to solve this problem with the specified constraints, as seen in Figure 10 .

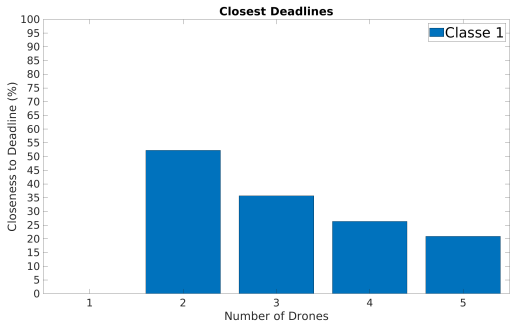

Fig. 10. Closest Visit to the Deadline in in a large AoI

C. Fleet Performance in AoIs with Special Cells

The presence of Special Cells requires at least 2 UAVs to guarantee that all Deadlines are respected. In these tests, some Cells are identified as Special Cells. As an example, a large AoI is represented in Figure 11. The Deadline ranges in this AoI are in Table I.

For the tested AoIs, at least 2 UAVs are always needed to prevent the Deadlines from expiring. In some cases, for larger AoIs, even if 2 UAVs are enough the results show that there is at least one Cell that is visited with the Deadline too close to expire. This is presented in Figure 12. In these cases, using more UAVs would be recommended. Besides, external events like wind effect can cause some visit delay and the UAVs will arrive to the Cell after the Deadline expired.

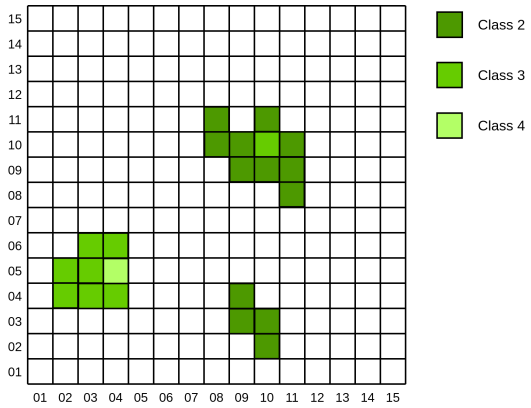

Fig. 11. Large AoI with Special Cells

Class	Ranges of Deadlines [s]
0	>1125
1	900.1 - 1125
2	675.1 - 900
3	450.1 - 675
4	225.1 - 450
5	0.1 - 225

TABLE I
RANGES OF DEADLINES IN EACH CLASS FOR A DETERMINED DEFAULT DEADLINE IN A LARGE AOI

D. Fleet Performance in AoIs with Special Cells

In these experiments, besides Special Cells, there are also some Restricted Cells that increase the Complexity of the AoI. The AoI in Figure 13 is an example of these AoIs. With the example it is possible to verify that when increasing the complexity of the AoI, a larger fleet of UAVs is needed. There are cases where, to ensure robustness in the solution, more UAVs than the amount tested are needed.

E. Alternatives for Unfeasible Problems

An unfeasible scenario is when, for a certain AoI, the Iterative Algorithm is incapable of finding a solution for the problem. Failing to find a solution is usually caused by using too few UAVs given the time constraints of the Cells

Two alternatives are explored: relaxing the Deadlines of Class 1 Cells and changing the velocity of the drones

1) Relaxing Deadlines of Class 1 Cells: Relaxing the Deadlines is an attempt to facilitate the access to Special Cells by the drones. Class 1 Deadlines, of the lowest priority, can be sacrificed if it would mean that Special Cells were to be visited earlier.

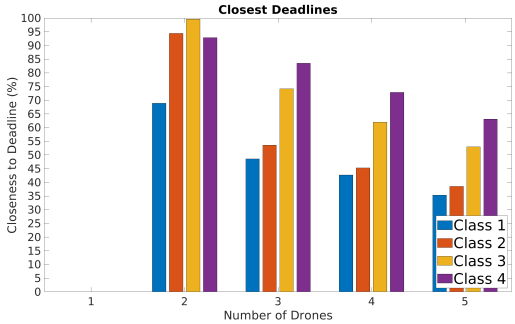

Fig. 12. Closest Visit for all classes with every number of UAVs

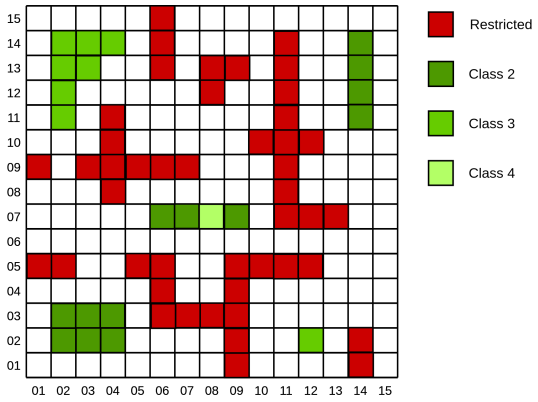

Fig. 13. Large AoI with Special and Restricted Cells

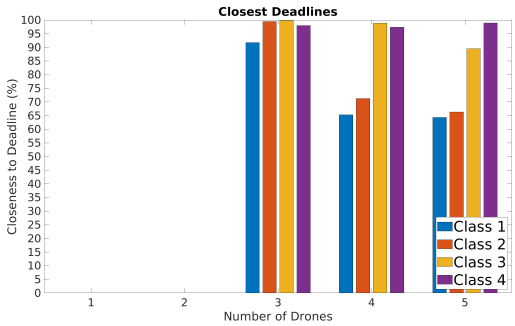

Fig. 14. Visit closest to the Deadline in every Class in a large AoI with Special and Restricted Cells

For the AoI in Figure 15, Class 1 Deadlines were increased gradually in amounts from 10 to 120 seconds. These Deadlines originally corresponded to the AoI's Default Deadline for the AoI.

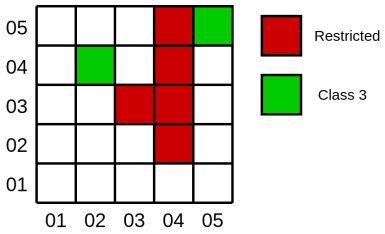

Fig. 15. Small AoI with Special and Restricted Cells

When using the Deadline that caused unfeasible solutions (125 seconds), the results were the ones presented in Figure 16 After increasing the deadline in 20 seconds, the results showed that the problem was feasible with 2 UAVs, as seen in Figure 17.

As a conclusion, changing the Deadline of Class 1 cells doesn't influence the order in which the Cells are visited, resulting in the same paths. The swaps in the Iterative Algorithm, if there are any for a given problem, are not affected by these changes. By changing these Deadlines, 1 UAV is never enough because of how close at least one visit to a Class 3 Cell is, using 2 UAVs.

2) Changing Drone Velocity: The constant t_{flying} controls the velocity of all UAVs. The results with the original velocity for the drones is shown in Figure 18. When the velocity is increased, the results are as seen in Figure 19.

Changing the drones velocities influence all Cells. This was already expected, since all drones will arrive to the Cells earlier. In terms of the Calendar of Operations for each drone, the Iterative Algorithm

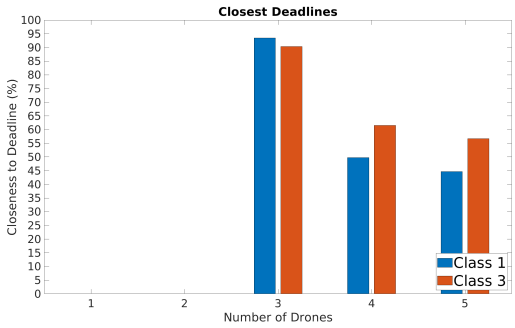

Fig. 16. Visit closest to the Deadline in every Class in a small AoI using the original Deadlines for Class 1 Cells

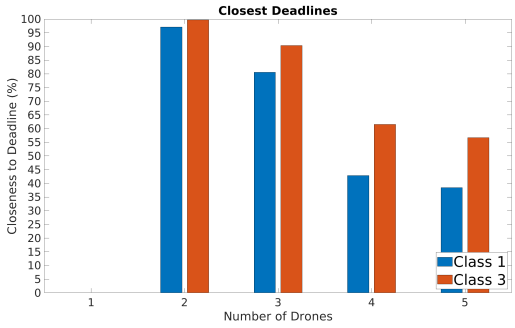

Fig. 17. Visit closest to the Deadline in every Class in a small AoI using Deadlines for Class 1 Cells 20 seconds higher

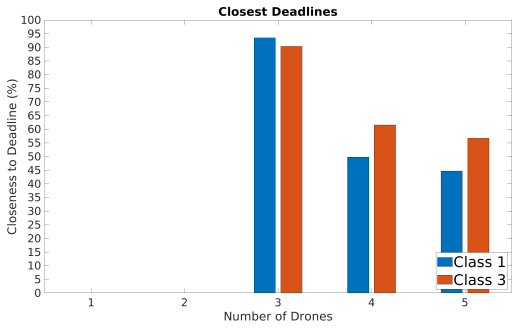

Fig. 18. Visit closest to the Deadline in every Class in a small AoI using the original UAV velocity

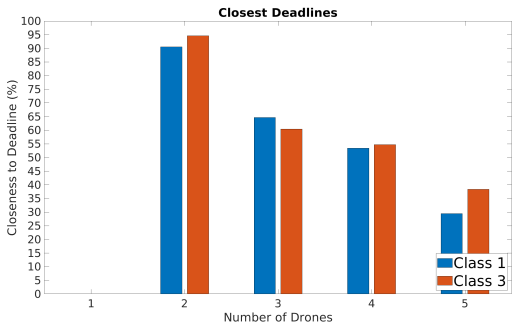

Fig. 19. Visit closest to the Deadline in every Class in a small AoI using the original UAV velocity

might need fewer changes to solve the problem for each AoI. However, this solution comes with higher energy consumption.

VII. Conclusion

In this dissertation, it is addressed a periodic Area Monitoring Problem. In this kind of problems, the usual goal is to find the best trajectories for UAVs such that the whole area to be monitored is visited given a set of constraints and maximizing one or more metrics. In this work, we tried to calculate the best trajectories to visit all regions of an area as fast as possible, guaranteeing that these are visited before the expiration of Deadlines imposed by the user.

We made some assumptions:

- The Area of Interest can be of various sizes.
- There can be deployed as many UAVs as points to sense in the area.
- The Cells of the AoI can be Restricted Cells (or obstacles) or belong to one of five Classes.
- Cells belonging to a Class higher than Class 1 are considered Special Cells, which have more demanding Deadlines that control the frequency in which all Cells are visited.

An Iterative Algorithm was developed to determine the trajectories for each UAV that assure that all the constraints of the problem are respected. This work was divided into two parts: First, an initial path is created; Second, the initial solution is improved until the goals are met. Two methods to generate the initial path were studied: Baseline and TSP.

A tool was implemented in MATLAB that not only models a problem of a particular area of interest but also solves the problem. The MATLAB tool implements the described methods and algorithms to find feasible solutions for every AoI and any group of UAVs. The solutions produce a path for every UAV that guarantees that every Deadline is respected. Using the tool, it's also possible to identify scenarios that are unfeasible.

From the results of this dissertation, some conclusions can be made. In most cases, choosing the TSP to generate the initial paths is better than the Baseline. This is because the TSP can be easily adapted to any AoI. In large AoIs of low complexity, generating the Baseline paths is as good as the TSP but consumes much less time. In real cases, with more complex areas, the need of more than 1 UAV is more evident. Another conclusion is that, although the number of UAVs always improving the closest visits to a Cell's Deadline, in the studied cases it stops being beneficial to use more UAVs.

There are some limitations to the presented work that would be interesting to explore in future work. The Baseline path, being an iterative solution, can be used in any AoI. Generating this path is only limited by the used hardware. The TSP, however, has a limited size for the AoIs where it is feasible to be used. That limit is an AoI with a size of about M = 30. In larger AoIs, generating this path takes a long time. Some inefficient visits were noted in several solution given by the Iterative Algorithm. Ignoring such visits, given the observed results, wouldn't compromise the problem's feasibility.

A Simulation environment is being considered to apply the found results. Using the simulator, the solutions provided by the algorithm can be tested and a more realistic environment. Factors like UAV velocity changes and time spent changing the height at which the drones fly to avoid collisions were not considered in the dissertation but can be tested in the simulation. These factors can influence the time when the drones reach a given Cell. In some cases, such delays can cause a Deadline to expire. Other factors, like the effect of wind and energy limitations, can also be considered in the Simulations. In the future, it would be interesting to test the results in a field environment, using real UAVs.

REFERENCES

[1] L. R. Pinto, L. Oliveira, L. Almeida and A. Rowe, "Extendable Matrix Camera Using Aerial Networks," 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2016, pp. 181-187, doi: 10.1109/ICARSC.2016.38.

[2] Alhalabi, W., Kitanneh, O., Alharbi, A. et al. Efficient solution for finding Hamilton cycles in undirected graphs. SpringerPlus 5, 1192 (2016). https://doi.org/10.1186/s40064-016-2746-8

[3] J. Jiang, X. Yao, E. Yang, J. Mehnen and H. Yu, "An Improved Adaptive Genetic Algorithm for Mobile Robot Path Planning Analogous to the Ordered Clustered TSP," 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1-8, doi: 10.1109/CEC48606.2020.9185672.

[4] J. L. Fargeas, B. Hyun, P. Kabamba and A. Girard, "Persistent visitation under revisit constraints," 2013 International Conference on Unmanned Aircraft Systems (ICUAS), 2013, pp. 952-957, doi: 10.1109/ICUAS.2013.6564781.

[5] Bektas, Tolga. "The multiple traveling salesman problem: an overview of formulations and solution procedures." omega 34.3 (2006): 209-219.

[6] Lu, Li-Chih, and Tai-Wen Yue. "Mission-oriented ant-team ACO for min–max MTSP." Applied Soft Computing 76 (2019): 436-444.

[7] Dimitris Dedousis and Vana Kalogeraki. 2016. "Complete Coverage Path Planning for arbitrary number of Unmanned Aerial Vehicles." In Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '16). Association for Computing Machinery, New York, NY, USA, Article 5, 1–4. https://doi.org/10.1145/2910674.2910719

[8] Kapoutsis, A.C., Chatzichristofis, S.A. and Kosmatopoulos, E.B. DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning. J Intell Robot Syst 86, 663–680 (2017). https://doi.org/10.1007/s10846-016-0461-x

[9] Mersheeva, V., Friedrich, G. (2015). Multi-UAV Monitoring with Priorities and Limited Energy Resources. Proceedings of the International Conference on Automated Planning and Scheduling, 25(1), 347-355. Retrieved from https://ojs.aaai.org/index.php/ICAPS/article/view/13695

[10] Y. Elmaliach, N. Agmon and G. A. Kaminka, "Multi-Robot Area Patrol under Frequency Constraints," Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 385-390, doi: 10.1109/ROBOT.2007.363817.

[11] M. Ahmadi and P. Stone, "A multi-robot system for continuous area sweeping tasks," Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., 2006, pp. 1724-1729, doi: 10.1109/ROBOT.2006.1641955.

[12] C. Liu, H. Du and Q. Ye, "Sweep Coverage with Return Time Constraint," 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1-6, doi: 10.1109/GLOCOM.2016.7842310.