UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RODRIGO SAAR DE MORAES

A DISTRIBUTED COOPERATIVE MULTI-UAV COORDINATION SYSTEM FOR CROWD MONITORING APPLICATIONS

Porto Alegre
2018

RODRIGO SAAR DE MORAES

A DISTRIBUTED COOPERATIVE MULTI-UAV COORDINATION SYSTEM FOR CROWD MONITORING APPLICATIONS

Thesis presented to Programa de Pós-Graduação em Engenharia Elétrica of Universidade Federal do Rio Grande do Sul in partial fulfillment of the requirements for the degree of Master in Electrical Engineering.
Minor: Automation and Control Systems

ADVISOR: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre
2018

RODRIGO SAAR DE MORAES

A DISTRIBUTED COOPERATIVE MULTI-UAV COORDINATION SYSTEM FOR CROWD MONITORING APPLICATIONS

This thesis was considered adequate for obtaining the degree of Master in Electrical Engineering and approved in its final form by the Advisor and the Examination Committee.

Advisor: _____
Prof. Dr. Edison Pignaton de Freitas, PPGEE-UFRGS

Examination Committee:

Prof. Dr. Walter Fetter Lages, PPGEE-UFRGS
PhD from Instituto Tecnológico de Aeronáutica (ITA) – São José dos Campos, Brazil

Prof. Dr. Luis Álvaro de Lima Silva , UFSM
PhD from University College London, UCL – London, UK

Prof. Dr. Bruno Castro da Silva, INF-UFRGS
PhD from University of Massachusetts Amherst – Amherst, US

Coordinator of PPGEE: _____
Prof. Dr. Valner João Brusamarello

Porto Alegre, March 2018.

ABSTRACT

Observing the current scenario, where terrorism and vandalism acts have become commonplace, particularly in big cities, it becomes clear the need to equip law enforcement forces with an efficient observation method, capable of identifying and observing potentially threatening individuals on crowds, to avoid or minimize damage in case of attacks. Moreover, with the popularization of small lightweight Unmanned Aerial Vehicles (UAVs), these have become an affordable and efficient tool, which can be used to track and follow targets or survey areas or buildings quietly, safely and almost undetectably.

This work presents the development of a multi-UAV based crowd monitoring system, demonstrating a system that uses small Commercial Of The Shelf (COTS) UAVs to periodically monitor a group of moving walking individuals. The goal of this work is to develop a coordination system for a swarm of UAVS capable of continuously monitoring a large group of individuals (targets) in a crowd, alternately observing each of them at a time while trying to not lose sight of any of these targets. A system equipped with a group of UAVs running this proposal can be used for law-enforcement applications, assisting authorities to monitor crowds in order to identify and to follow suspicious individuals that can have attitudes that could be classified as vandalism or linked to terrorist attack attempts.

To address this problem a system composed of three parts is proposed and was developed in this thesis. First, an auction algorithm was put in place to distribute interest targets among the multiple UAVs. The UAVs, in turn, make use of a genetic algorithm to calculate the order in which they would visit each of the targets on their observation queue. Moreover, a target handover algorithm was also implemented to redistribute targets among the UAVs in case the system judged that a target was about to be lost by its current observer UAV.

The proposed system was evaluated through a set of experiments set-up to verify and to demonstrate the system capabilities to perform such monitoring task, proving its efficiency. During these experiments, it is made clear that the system as a whole has a great potential to solve this kind of moving target monitoring problem that can be mapped to a Time Dependent Travel Salesman Problem (TDTSP), observing targets, and redistributing them among UAVs as necessary during the mission.

Keywords: Multiple-UAV systems, target tracking, crowd monitoring, UAV monitoring, multi-target observation, robot coordination.

RESUMO

Ao observar a situação atual, na qual atos de vandalismo e terrorismo tornaram-se frequentes e cada vez mais presentes ao redor do mundo, principalmente em grandes cidades, torna-se clara a necessidade de equipar as forças policiais com tecnologias de observação e monitoramento inteligentes, capazes de identificar e monitorar indivíduos potencialmente perigosos que possam estar infiltrados nas multidões. Ao mesmo tempo, com sua recente popularização, veículos aéreos não tripulados, também chamados VANTs e conhecidos popularmente como "drones", acabaram por tornar-se ferramentas baratas e eficientes para diversas aplicações, incluindo observação, fornecendo a seus utilizadores a capacidade de monitorar alvos, áreas, ou prédios de forma segura e quase imperceptível.

Unindo estas duas tendências, este trabalho apresenta o desenvolvimento de um sistema multi-VANT para observção de alvos móveis em multidões, demonstrando a possibilidade de utilização de pequenos VANTs comerciais comuns para o monitoramento de grupos de pedestres. O principal objetivo de tal sistema é monitorar continuamente indivíduos de interesse em um grupo de pessoas, visitando cada um destes indivíduos alternadamente, de forma a manter um registro geral do estado de cada um deles. Um sistema deste tipo poderia, por exemplo, ser utilizado por autoridades no controle de manifestações e outras atividades em que grandes grupos de pessoas estejam envolvidos, ajudando a polícia e outros órgãos a identificar indivíduos com comportamento suspeito ou agressivo mais rapidamente, evitando ou minimizando os efeitos de atitudes de vandalismo e de ataques terroristas.

Com o intuíto de abordar tal problema da forma mais completa e adequada possível, esta tese apresenta a concepção e o desenvolvimento de um sistema híbrido composto de três diferentes algoritmos: um algoritmo de distribuição de alvos; um de roteamento; e um de repasse de alvos. Primeiramente, neste sistema, um algoritmo de distribuição de alvos baseado em um paradigma de mercado que simula um leilão distribui os alvos entre os VANTs da melhor forma possível. Os VANTs, por sua vez, utilizam um algoritmo genético de roteamento para resolver uma instância do Problema do Caixeiro Viajante e decidir a melhor rota para visitar cada alvo sob sua responsabilidade. Ao mesmo tempo, o sistema analisa a necessidade de redistribuição dos alvos, ativando um algoritmo capaz de realizar esta ação ao perceber sua necessidade quando na iminência de perder algum alvo de vista. Ao fim de seu desenvolvimento, o sistema proposto foi testado em uma série de experimentos especialmente desenvolvidos para avaliar seu desempenho em situações controladas e comprovar sua eficiência para realizar a missão pretendida.

Palavras-chave: sistemas multi-vant, monitoramento de alvos, sistema de coordenação de robôs, monitoramento de pessoas, monitoramento com vants.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

A-TSP	Asymmetric Travelling Salesman Problem
B-CMOMMT	Behavioral Cooperative Multi-Robot Observation of Multiple Moving Targets
CMOMMT	Cooperative Multi-Robot Observation of Multiple Moving Targets
CSAT	Cooperative Search, Acquisition and Tracking
CT	Cooperative Tracking
CNP	Contract Net Protocol
FOV	Field of View
GA	Genetic Algorithm
HW	Hardware
MPE	Multi-Robot Pursuit Evasion
MT-TSP	Moving Target Travelling Salesman Problem
RF	Radio Frequency
S-TSP	Symmetric Travelling Salesman Problem
SW	Software
TD-TSP	Time dependent Travelling Salesman Problem
TSP	Travelling Salesman Problem
TSP-NR	Travelling Salesman Problem without Repetitions
TSP-R	Travelling Salesman Problem with Repetitions
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VM	Virtual Machine

CONTENTS

1 INTRODUCTION

1.1 Motivation

The usage of Unmanned Aerial Vehicles (UAVs) in civilian and military applications has increased exponentially since cheaper and more complete unmanned aircraft platforms have appeared on the market. This trend, allied to other technological developments in both software and hardware fields has originated a great number of new applications and new uses for this kind of technology (KELLER, 2015).

Besides that, micro UAVs have become an important tool to law enforcement forces, allowing them to track and follow targets or survey areas or buildings quietly, safely and almost undetectably. These kinds of applications, however, depend today heavily on human control and supervision, increasing costs and complexity to perform missions and making them prone to human mistakes that would not occur if an autonomous control system was in place. Among these uses, crowd control and monitoring applications for law enforcement present a set of interesting and complex characteristics that turn the development of an autonomous coordination for such systems in these application scenarios a challenging task.

The importance of such systems becomes clear once the statistics show that more than 141,000 terrorist attacks, causing more than 706,000 causalities, were committed from 1970 to 2014 (LAFREE; DUGAN, 2017), among which 25% were aimed at the civil population directly. Since most of these attacks consisted of bomb attacks (48.5%) and 25.7% consisted on armed attacks it is reasonable to suppose that an efficient observation method, capable of identifying and observing potentially threatening individuals on a crowd, could assist law enforcement in avoiding or minimizing the damage caused by the attackers on these situations.

In general, this type of problem presents more targets, or groups of targets, in the crowd than resources to monitor them. While demonstrations may be composed of tenths of thousands of individuals, police forces and resources are often restricted to hundreds (or few thousands in the best case) assets, restricting the capacity of law enforcement agencies to act in such situations. This fact creates a need for intelligent monitoring solutions in order to assist the law enforcement forces. In order to address this situation and the related problem, an intelligent and autonomous system of crowd monitoring must be able to identify potentially dangerous individuals in the crowd, e.g. individuals using masks or carrying potentially dangerous objects, and track them individually. Usually, a system of this kind operates by allocating some of the agents to a specific individual while maintaining other agents or resources monitoring the crowd as a whole to find new potential threats or risky situations.

Observing this landscape, this work investigates the usage of UAVs in this kind of

crowd monitoring and control operations and presents a system capable of using multiple UAVs in this context. The system is able to to find targets in the crowd, to distribute target responsibilities among the UAVs, and to effectively monitor all the targets, minimizing the time each of them remains hidden from the system. As previously stated, the system has to be able to work in situations in which there are more targets than UAV assets, making sure that it is able to handle a great number of targets, if necessary.

1.2 Contribution

Most multi-UAV multiple target observation systems presented in the literature focus themselves in spatial clustering or target grouping, allocating each UAV to a specific portion of terrain or group of targets and, mostly, forcing them into orbital or circular flight paths. This kind of approach may be a good fit to larger fixed-wing UAVs, but might not be the best approach once a small, cheap and lightweight UAV that flies closer to the targets, covering smaller portions of ground, but possibly in urban environments, is considered. Small cheap commercial-off-the-shelf (COTS) UAVs can, for example, be flown in urban environments, such as avenues and parks, or even indoors, as in sports gymnasiums or conference centers, presenting themselves as affordable, dynamic, and efficient assets that could be explored by law-enforcement forces for monitoring purposes.

Considering this scenario, this work proposes a novel coordination technique that explore the characteristics of small COTS UAVs to monitor multiple moving targets from a closer range. Besides that, this work presents a cooperative communication behavior, rehandling and auctioning targets to other UAVs when a sight loss is imminent, a different approach than most cooperative tracking and observing techniques, that usually handle target attributions as fixed throughout mission time. Consequently, the dynamic approach here proposed improves the overall performance of the system. It is also important to notice that this work is mainly focused on the coordination of the UAVs, leaving aside problems like movement control and image processing, which can be very complex on their fields, but that are out of the scope of this work and then not addressed here.

Thus, it is possible to state that the contribution of this thesis is mainly two-fold: First, the presentation of a suitable model to handle the problem of crowd monitoring with a group of small UAVs, characterizing it as a Time dependent Travel Salesman Problem (TDTSP). Second, the presentation of an efficient solution that distributes the targets among the UAVs and that is able to keep track of them under varying conditions during the surveillance mission.

1.3 Structure

Section 2 of this work presents the theoretical background used as a basis in the implementation, idealization, test and analysis of this work. This section covers themes as Multi-Robot Observation of Moving Targets, Human Mobility Models and the common Travelling Salesman and Task Assignment problems, all very relevant and related to the work presented on this thesis. Section 3, in turn, presents similar problems and approaches found in the literature, collaborating to set the context and contributions of the work developed here.

While Section 4 is reserved to the problem definition, presenting its properties, its assumptions and the characteristics that make it a challenge, Section 5 is focused on the presentation and detailing of the many methods and algorithms that compose the whole

solution, outlining techniques and explaining how each part of the problem is solved. Section 6, for its part, brings the experiments and their results, explaining the experimentation environment and how the experiments themselves were set-up.

Section 7 is reserved for the final considerations, discussing the results of the overall process and proposing further works that may complement the proposed system.

2 BACKGROUND

2.1 Multi-Robot Multiple Moving Target Observation

The usage of robots to observe and monitor multiple moving or static targets is an important area of interest for many applications. Robots can be used, for example, to monitor demonstrations or other urban activities, to capture sport events live feeds (NATALIZIO et al., 2013), to survey civil or military areas (MORAES; FREITAS, 2017), or to simply follow wildlife for research purposes (LINCHANT et al., 2015). This trend has increased exponentially since cheaper and more complete robotic platforms, i.e. UAVs, have appeared on the market, originating a great number of new applications and new uses for this kind of technology. However, controlling and coordinating groups of robots to monitor targets is no easy task. A system of cooperative robots that observe multiple targets requires a control technique capable of combining target distribution, coordination, and motion planning approaches working together to maximize the number of targets under observation.

Coordination of robotic-based systems is a widely studied research topic in robotics an automation literature. Most algorithms focus themselves in spatial clustering and ground monitoring, allocating each robot to a specific portion of the monitored terrain and giving it the exclusivity responsibility for the targets present on that area. Variations of this approach allocate a group of targets to a given robot, which ends up having the same mobility restrictions. Coordinating mobile robots to monitor moving targets also depends greatly on the communication resources available to the robots. While some robots, such as large UAVs or UGVs, may support complex and power demanding communication hardware, light commercial-off-the-shelf (COTS) UAVs, such as quad-copters, present an affordable solution for smart-surveillance application, but cannot support power hungry hardware, limiting the communication range and capabilities of such assets. In addition to that, there is also the aspect of whether the coordination should be centralized, decentralized or distributed, inserting or not points of failure on the system and making it, or not, capable of taking self-healing actions. Many of these different approaches are analyzed in (KHAN; RINNER; CAVALLARO, 2018), which organizes, categorizes and compares 20 years of studies in the usage of mobile robots for observing multiple moving targets.

Expanding the idea of (KHAN; RINNER; CAVALLARO, 2018), the problem of observing multiple targets using a group of cooperative can be defined and formulated as follows:

Definition 1 (Multi-Robot Observation of Multiple Moving Targets)**.** *Given a set of n targets $T = \{t_1, ..., t_n\}$ moving independently on a given environment, the state of target t_i at time t can be defined in terms of its position (x,y and z coordinates) as follows :*

$$\chi_i^t = (x_i, y_i, z_i)^t \tag{1}$$

and a set of m *robots* $R = \{r_1, ..., r_m\}$ *on the same environment, whose state is characterized by*

$$\rho_j^t = (x_j, y_j, z_j)^t \tag{2}$$

The observation O_{ij}^t *of target* t_i *at time* t *by robot* r_j*, which has a sensor with FOV* F_j*, is defined as follows:*

$$O_{ij}^t = \begin{cases} 1; & \textit{if} \quad \chi_i^t \in F_j^t \\ 0; & \textit{otherwise} \end{cases} \tag{3}$$

The problem is to plan the motion of every $r_j \in R$ *in order to maximize the observation of all the targets during the observation period* τ*, or, mathematically:*

$$\underset{O_i}{\textit{maximize}} \quad O_i = \int_0^\tau \sum_i^n \sum_j^m O_{ij}^t \, dt \tag{4}$$

Observing this definition it can be assessed that a series of different variables can influence the outcome of an observation mission. The relation between the number of robots (m) and the number of targets (n) is one of these variables. In case $m \gg n$, it is very likely that the drones will be able to observe and track the targets constantly without losing them, while, for $m \ll n$, the same cannot be said, since there are less robots monitoring the targets than targets being monitored. The problem becomes even more challenging when n or m are not constant, meaning that targets or robots may appear or disappear during the mission, imposing more challenges to the path planning algorithms, and even, in the worst case, leading to inconsistencies in distributed systems.

2.1.1 Characterization of Multiple Target Observation Scenarios

According to (KHAN; RINNER; CAVALLARO, 2018), there exist five main factors that characterize and influence every single multiple moving target observation scenario: the environment in which the observation takes place; the targets being observed; the robots observing the targets; the sensing equipment; and the coordination method.

2.1.1.1 Environment

The first important factor is the environment, which can be defined as the setting or conditions in which the observation mission is carried on. The most important characteristic of environments is their structure, which can restrict mobility for both UAVs and targets, in case obstacles exist, and sensor coverage, in case the targets are able to hide behind these obstacles. Environments can be classified into two categories considering their structure: Structured Environments, which provide information on the presence of obstacles and boundaries; and Unstructured Environments, which provide no information whatsoever. There are also a third type, the evolving environments, which present a dynamic structure that can change during the mission.

2.1.1.2 Targets

Along with the environment, the targets are another factor worth mentioning. In most applications, the mobility of targets is linked to the environment structure, such as the presence of roads or paths. This movement is also usually unforeseen, and sometimes unpredictable, depending on the problem. Another common assumption is that targets move independently from one another, meaning that one target behavior does not influence or provide insight on how others move

Targets can also be classified, according to their level of cooperation with the robots, in three categories:

1. Cooperative Targets: Are those targets that periodically, or continuously, broadcast some information about themselves to the robots, i.e. location ;

2. Non-Cooperative Targets: The Non-Cooperative group is constituted by targets that neither communicate with the robots, nor try to evade observation.

3. Evasive Targets: Are the targets that try to hide, or escape from being observed, presenting a certain challenge to the observation mission

2.1.1.3 Coordination

The last factor worth mentioning is the coordination strategy. The coordination of a group of mobile robots is highly dependent on a reliable communication network through which they can share information with each other. This exchange is vital to the mission success, and any type of disconnection or failure may cause the system performance to degrade significantly.

As for the organization, coordination can be centralized, decentralized, or distributed. Centralized coordination means that robots exchange information with a central node, which processes it and coordinates the robots. This type of approach, however, introduces a single point of failure that can severely affect the whole system. Besides, it hugely dependent on the communication, that cannot be very reliable since robots are moving around on the environment. Decentralized algorithms, however, are less susceptible to failures once local leaders coordinate small groups of robots. Distributed algorithms, on the other hand, do not rely at all on any type of leadership. Robots in distributed environments take their own decisions with the information they have at the moment, even if it is limited. Distributed approaches, however, require more processing power and memory, resources that may not be available to all the robots.

2.1.2 Classes of Multiple Target Observation Problems

(KHAN; RINNER; CAVALLARO, 2018) also groups the existing approaches on four major control techniques based on how they deal with the mission and on their objectives. These four techniques are discussed below:

1. Cooperative Tracking (CT): is a control technique whose objective is reducing the time duration between two observations to the same target, updating the information the system has on the location of the targets on a more frequent basis. In this kind of approach, the robots either know where the targets are or are able to predict their location, not searching for new targets during the mission. Most approaches in the literature are based on this technique once it allows the responsibility of a robot over

a target to be defined at early stages, making only path planning activities necessary during the missions.

2. Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT): this technique is focused on maximizing the collective time of observation for all targets. CMOMMT differs from CT once, instead of trying to keep track of the targets positions through successive visits, i.e. occasionally leaving the target and later returning back, it tries to maximize the time a target effectively stays into the field of view of the UAV. This kind of approach usually is composed of two states, search and track, the first aims on finding targets when no target is under its FOV, while the second in pursuing them. To deal with the different aspects of CMOMMT, some variants were developed in the literature to better fit some problems requirements:

   (a) A-CMOMMT(Approximate): created to reduce the observation overlap of a single target by multiple robots.

   (b) B-CMOMMT (Behavioural): robots implement a "help" call that asks other robots for help when on the imminence of losing a target. Also introduces target tagging techniques to reduce observation overlap.

   (c) F-CMOMMT (Formation): uses a flexible formation of robots to fulfill the goals of CMOMMT.

   (d) P-CMOMMT (Personal): analyzes the solution information to infer data about the observation diversity. This specific variant addresses a problem of the generic CMOMMT formulation, where some targets may be under observation almost the whole time, while others are not observed at all.

   (e) W-CMOMMT (Weighted): assigns weights to the targets, minimizing the problem of losing targets during the missions.

3. Cooperative Search, Acquisition and Tracking (CSAT): The goal of CSAT is to search for new targets and track the existing ones. The switching between search and track modes is usually based on the time or the level of uncertainty concerning a given target location. The idea behind this approach is giving the robots the opportunity to search for new targets, while maintaining the level of uncertainty about a given target location under a limited threshold.

4. Multi-Robot Pursuit Evasion (MPE): MPE approaches are those in which targets are evasive. The role of robots in MPE scenarios is to capture targets once, minimizing the time required to capture one or more evasive targets.

2.2 Travelling Salesman Problem

2.2.1 The General Travelling Salesman Problem

The Traveling Salesman Problem (TSP) (APPLEGATE, 2006) is an important problem in computer science which consists in finding a path linking a set of cities so that each of then can be visited once. In other words, the general TSP consists in finding the shortest path to visit a set of cities exactly once and return to the initial point (first visited city). The problem can be formulated as:

$$\begin{aligned} &\underset{D_t}{\text{minimize}} \quad D_t = \sum_{i}^{n} \sum_{j}^{n} c_{ij} d_{ij} \\ &\text{s.t.} \qquad\qquad \forall i \neq j \end{aligned} \tag{5}$$

$$\text{where} \quad c_{ij} = \begin{cases} 1;\ \text{if salesman travels from city i to city j} \\ 0;\ \text{otherwise} \end{cases} \tag{6}$$

and d_{ij} is the distance between cities i and j. A possible solution of the TSP, forms a tour by visiting each city at least once and can be represented as a set of n ordered pairs of cities as $S = \{(1,2),(2,5),(5,3),(3,4),(4,1)\}$. In which each element (i,j) refers to a path between cities i and j of the tour. In other words, the primary objective is to find the permutation of cities which minimizes the total distance.

TSP's specializations and generalizations have been a common and important topic of study for both mathematicians and computer scientists over the years as many problems can be reduced to the TSP (MATAI R; MITTAL, 2010) With it's first mentions in the literature dating back to the nineteenth century, this routing problem has evolved with the technology around it. Nowadays TSP's are found as sub-problems of several other more general applications such as transportation infrastructure planning, logistics, Printed Circuit Board projects, DNA sequencing, communication routing and even military intervention planning (MATAI R; MITTAL, 2010). These are just a small enumeration of the problems that are mapped to and need to deal with TSP in their solutions.

A great number of different approaches for solving the TSP has been proposed during recent decades (MATAI R; MITTAL, 2010; RAMAN; GILL, 2017; ARRAM; AYOB; ZAKREE, 2014). Most of these methods can be divided in heuristic (ARRAM; AYOB; ZAKREE, 2014) and exact methods. Exact methods search for the optimal solution and can only optimally solve small problems, since their approaches result in exponential computational complexities for large problems. Heuristic methods, however, trade optimality for speed, being satisfied by local optimal solutions presenting good results in solving large problems.

Recently, the usage of new technological assets such as Unmanned Aerial Vehicles (UAVs) and autonomous agents such as robots or autonomous vehicles (including autonomous UAVs) has exponentially increased since cheaper and more complete autonomous platforms have appeared on the market. This trend, allied to other technological developments in both software and hardware fields has originated a great number of new applications, some of which need to deal with TSP-like problems.

2.2.2 Variants of the Travelling Salesman Problem

As discussed on the previous section, the importance of the TSP is that it can be applied on different real-world applications, i.e. logistics, transportation and DNA sequencing, helping with multiple important problems in different areas of science. real-world problems, however, have their own sets of constraints that often cannot be met by the general formulation or solution of the TSP. Thus, the traditional model of the TSP has been modified by several researchers to match different problems and its constraints (ILAVARASI; JOSEPH, 2014; GUTIN; PUNNEN, 2002).

The first basic variants of the problem are known as symmetric TSP (sTSP), and asymmetric TSP (aTSP), and are related to the symmetry of the problem. They can be defined as follows:

- S-TSP: is the most simple case, where the distance to travel from city i to city j is the same of going from j to i, meaning every path between two cities is symmetrical. Namely, $d_{ij} = d_{ji}$.

- A-TSP: is the case in which, for at least one pair (i, j) of cities $d_{ij} \neq d_{ji}$, meaning that at least one path between two cities is not symmetrical.

Similarly, there exist problems in which the uniqueness of visits is not required, meaning that to solve a problem, i.e. in logistics, a traveler has to visits a city more than one time. According to this criteria, the TSP problem can also be classified into two other variants, the TSP-NR, and the TSP-R, which are defined as follows:

- TSP-NR: is the general case, in which each city is visited exactly only once.

- TSP-R: is the case in which repetitions are allowed, meaning that a city may be visited more than once in a tour.

Sometimes, however, the constraints required by the problem are not so simple. In some cases, the cost of traveling between cities may depend on other factors, such as time. In other cases the problem requires the metric used to evaluate the solutions to be changed, evaluating the solutions based on a profit measure instead of distance, or even maximizing the overall tour distance instead of minimizing it. The most important variations, can, however, be grouped into four different classes of TSP problems, as presented below

- TSPTW: the Traveling Salesman Problem with Time Windows (TSPTW) consists in finding the minimum path to visit a set of nodes exactly once. A node, however, must be visited within a fixed time window such that if the vehicle arrives too early it has to wait until the window opens (again) to effectively mark that node as visited.

- PB-TSP: the Profit Based Traveling Salesman Problem are a set of problems in which solution are evaluated according to a profit metric that has to be maximized. In some cases, it is not even necessary to visit all cities to solve the PB-TSP problem.

- MAX-TSP: Is the problem where the salesman has to visit each of a set of cities exactly once and return to the starting city with maximum possible distance traveled. The exact opposite of the general case, in which the distance is minimized.

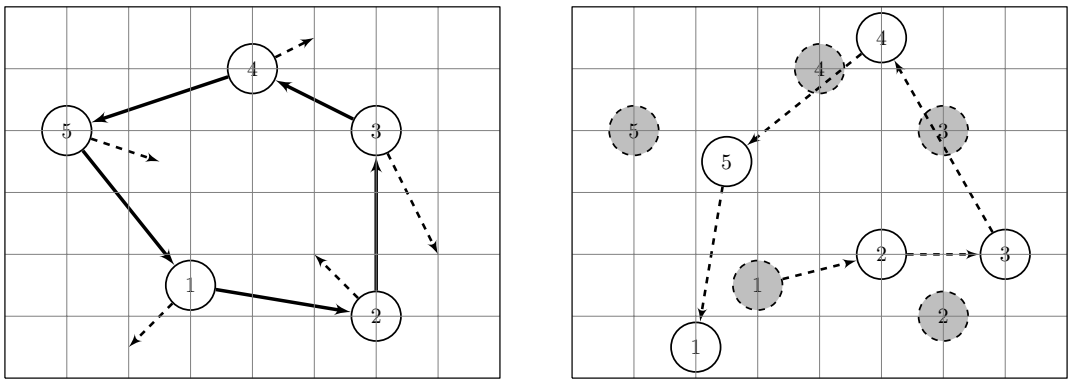
- TD-TSP: The Time-Dependent Traveling Salesman Problem (TD-TSP) is a generalization of the Traveling Salesman Problem (TSP), in which the cost to travel between cities depends and changes over time. These costs can either depend on the position of a city position in the tour, or on the movement of the cities, or nodes (K-TSP or MT-TSP)

2.2.3 Moving-Target TSP

The MT-TSP is a special case of the Time-Dependent Traveling Salesman Problem (TD-TSP) in which, not only the cost to travel between cities changes with the time, but the cities, or targets, in this case, move with a given speed as time passes. (HELVIG; ROBINS; ZELIKOVSKY, 2003) introduces this problem as a generalization of the Traveling Salesman Problem where targets can move with constant velocities, formulated as follows:

> Given a set $S = \{s_1, ..., S_n\}$ of targets, each s_i moving at constant velocity $\vec{v_i}$ from an initial position p_i, and given a pursuer starting at the origin and having maximum speed $v > |\vec{v_i}|$, find the fastest tour starting (and ending) at the origin, which intercepts all targets.

Figure 1 shows a graphical representation of an example of this problem. On the static case of the TSP the traveler, who is initially in city 1, calculates in instant t_0 the best path to solve the static problem (Figure 1a). The problem with MT-TSP is, however, that in instant t_1, after the traveler leaves departing city (node 1), all nodes have moved and changed their positions. This fact requires a new tour to enable the traveler to intercept the nodes after these movements, where they have moved to (Figure 1b). As the nodes keep moving, this process must be repeated until it returns to node 1, which is in a new position (completely different from the departing one) at the end of the tour.



(a) Instant t_0 (b) Instant t_1

Figure 1: An example of the Moving Target Traveling Salesman Problem

Source: Prepared by the author

In order to solve this particularly complex problem, multiple heuristic-based approaches have been proposed in the literature. For instance, the work in (CHOUBEY, 2013) presents a Genetic Algorithm approach to solve a set of TD-TSP problems. The results showed that the Genetic Algorithm implemented by the authors is more effective than solving the problem using a similarly modified greedy method, proving the advantages of the heuristic method to solve this kind of problem.

The work presented in (ABELEDO et al., 2013) proposes a polyhedral basis Branch-Cut-and-Price Algorithm that suggests polyhedral theory can play an important role in improving algorithms for the MT-TSP and TD-TSP. Performing studies on the TD-TSP polytopes their method proposes polyhedral cuts that can be used to modify the BCP

approach allowing the solution of 22 TSP instances with up to 107 vertices.

A repetitive heuristic solution of the traveling salesman problem is used in (ENGLOT; SAHAI; COHEN, 2013) to solve a TSP in which a large number of moving targets must be intercepted by a single agent as quickly as possible. The author's work compares the Lin-Kernigha heuristic to a greedy heuristic over different problem parameterizations showing that the benefits of a non-greedy solution depend on the speed of the targets relative to the agent. The work proves that LKH is superior to greedy heuristics when the targets are moving at low speeds and that its relative performance to greedy algorithms improves as sensor noise worsens.

2.3 Heuristic Methods Applied to TSP

2.3.1 Genetic Algorithm Methods

Genetic Algorithms (GAs) (HOLLAND, 1992) are a kind of evolutionary algorithms that try to simulate the concepts of natural Darwinian evolution to solve complex optimization problems. The concept behind these methods is to evolve an initial population of candidate solutions to a problem until one of the solutions on this pool or population is considered good enough to solve the desired problem.

The idea of Genetic Algorithms is the application of a Survival of the Fittest principle on a population of individuals representing potential solutions to the problem being solved. By using this principle it is granted that only the fittest individuals are able to survive and mate, causing potential solutions to evolve and become better and closer to the optimal solution without performing an extensive brute force search. As in real life, each individual on the algorithm has unique genetic code that defines it's phenotype and traits and rules the behavior of the specific candidate solution represented by each individual. Through a series of operations such as selection, crossover and mutation, the genetic code of the population is evolved for a certain number of generations, or iterations, until the algorithm decides to stop and selects the best solution among the pool of solutions it created.

This process of selecting and evolving a solution starts by selecting a representation of the problem as to make sure that each candidate solution has its own representation. An initial population of candidate solutions is then created, usually at random even though heuristics can be used for that. Each candidate of the population is analyzed, it's fitness, or how well the candidate solves the problem, calculated and the generation sorted according to fitness values. The Survival of the Fitness principle is then applied, meaning that most fit solutions have more chance of breeding generating offspring to be part the next generation. According to this principle, two parents among the candidates are then chosen, it's genetic material combined into a new genetic code, and an offspring created. Next, the mutation occurs. Some of these offspring are randomly selected to suffer mutations in its genetic code, creating new unique patterns on the population genetic pool. This process is repeated until a new generation is completely created. After that, the algorithm applies the Survival of the Fitness principle again on this new generation and all the crossover and mutation process repeats itself. The whole algorithm is iterated over and over until either a stop condition is found or the maximum number of generations exceeded. By the end of the whole process, the best solution to the problem is chosen amongst the pool of individuals created during the execution of the algorithm.

One of the most important characteristics of this kind of algorithm is its ability to avoid presenting poor local optima candidate solutions as definitive solutions until the algorithm

is finished. It is very important that worst fit individuals still have a chance to mate, and not be discarded, being able to create useful genetic material that may contribute to the generation of better solutions than local optimal ones.

For comprehension purposes, a graphical representation of the steps and workflow of a genetic algorithm solution can be seen in Figure 2

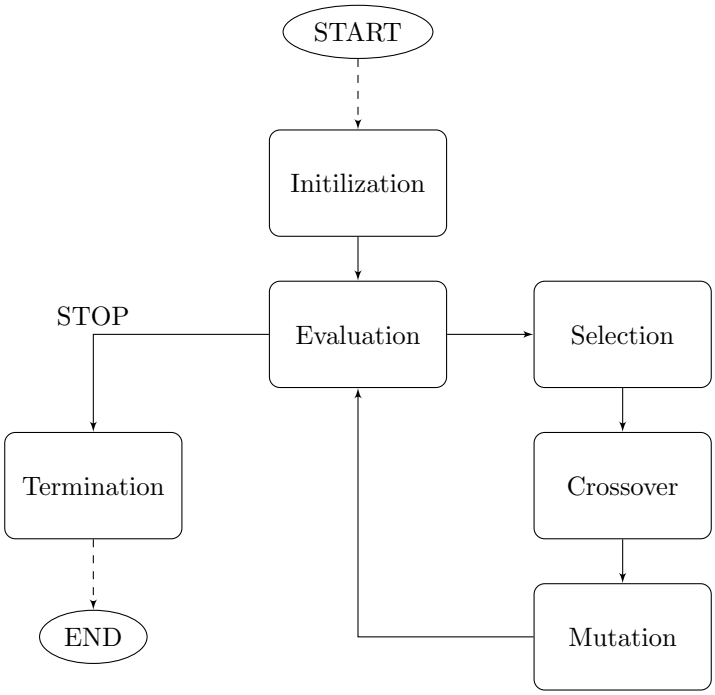


Figure 2: GA Fluxogram

Source: Prepared by the author

Besides its simple concept, "pure" genetic algorithms were not designed to solve combinatorial problems. The main intent behind the development of this type of algorithm in the 60's and 70's was the optimization of numerical functions (HOLLAND, 1992). Consequently, the original algorithm had to be modified to represent and handle combinatorial optimization problems like the TSP (POTVIN, 1996).

The first problem in using GAs for any kind of optimization problem is choosing the correct representation method (POTVIN, 1996) for the desired problem solution search space, ensuring that only valid and unique representations of each solution exist, and that a chosen representation method does not influence crossover, selection or mutation operations as to generate unsuitable results.

The second and third principal problems are related to the crossovers(POTVIN, 1996; ÜÇOLUK, 2002; GOLDBERG; LINGLE JR., 1985) and mutation phases of the algorithm. As previously illustrated on the text and in Figure 2, crossover and mutations operators are the methods responsible for creating new solutions for the search space, ensuring that new and possibly better solutions are examined each time a new generation is processed. Their choice plays, therefore, an important role on the convergence of the algorithm and on the quality or optimality of the solutions that can be found.

The fourth problem deals with the selection problem (SHUKLA; PANDEY; MEHROTRA, 2015; RAZALI; GERAGHTY, 2011) meaning what strategy is used to choose in-

dividuals from a previous population to be parents and generate the offspring that will compose the next generation.

The following subsections present a synthesis of the principal optimizations to genetic algorithms proposed on literature to improve their performance when handling TSPs. Since an extensive review of GA methods is not the primary focus of this work, the different representation methods, selection methods, crossover operators, and mutations operators will be briefly discussed and compared for informative purposes.

2.3.1.1 Representation Methods

One of the first problems faced when using GA for TSP purposes is choosing the correct representation for the search space. A suitable representation must ensure that only valid and unique representations of each solution exists, and that a chosen representation method does not lead to crossover, selection or mutation operation to unexpected or undesired behaviors. Since the TSP works on a combinatorial permutation search space, the problem becomes even more complex.

Considering a representation in which a tour is represented as a list of cities, i.e. (a,b,c,d,e,f). The first problem is that a tour representation, for example, is not unique, a same tour may have multiple representations. I.e. tour (a,b,c,d,e,f) is the same as (b,c,d,e,f,a). Besides uniqueness, validity is also a problem, since valid parents can generate invalid offspring after crossover, (a,b,c,d,e) and (a,d,e,c,b), for example, can generate (a,b,c,c,d) as an offspring, an invalid tour.

To solve these problems, different kinds of representations methods were studied in the literature, the three most common are cited below:

1. Adjacency Representation: The adjacency representation (GREFENSTETTE et al., 1985) represents a tour as a list of n cities. The city j is listed in the position i if and only if the tour leads from city i to city j. This representation is designed to facilitate the manipulation of edges. The crossover operators based on this representation generate offspring that inherit most of their edges from the parent chromosomes. This representation, however, does not allow a classical crossover operator to be used.

2. Ordinal Representation: The ordinal representation (GREFENSTETTE et al., 1985) represents a tour as a list of n cities in which each element is associated to the position of this city on a canonical tour, that serves as a reference point for lists in ordinal representations. The main contribution of this coding scheme is that one-point (classical) crossover always generates valid offspring.

3. Path Representation: Path representation is considered to be the most classical and natural representation, representing tours by a list of the cities ordered as they appear on the tour. The crossover operators based on this representation typically generate offspring that inherit the relative order of the cities. This representation, however, has to be checked for uniqueness, since each tour can be represented in $2n$ distinct ways because any city can be placed at position 1.

2.3.1.2 Crossover Operators

The crossover operator is perhaps one of the most important parts of a Genetic Algorithm. This operator is the module responsible for recombining parents' genetic code into offspring, creating new and different candidate solutions in the search space of the

optimization problem. Crossover is also the operation that has the most significant impact on the speed of convergence of a GA, influencing directly on how fast good or optimal solutions will be found.

Besides defining the convergence rate, the crossover has also a great impact on the quality of the solutions. A good crossover algorithm or operator can lead to quasi-optimal or optimal results that a poor operator will never achieve. On a TSP problem this influence becomes even more visible, once a population of valid parents has, by means of the operators, a high chance of generating invalid offspring, with some cities missing, and others repeated.

In order to avoid the generation of invalid offspring, which is the main issue with crossover on TSP a great variety of algorithms has been developed on literature. Solutions are commonly observed to fit into one of the following three categories: Disqualification; Repairing; and Specialized Operators (ÜÇOLUK, 2002). While Disqualification Techniques allow the algorithm to create invalid solutions that will later be disqualified, Repairing techniques take invalid solutions and try to repair them. Specialized Crossover Operators, in their turn, usually present the best results in terms of quality and convergence, avoiding invalid results and operating on the offspring as to only create individuals with feasible tours.

Among the several crossover techniques presented in the literature, some of the most important worth mentioning are:

1. Partially-Mapped Crossover (PMX): (GOLDBERG; LINGLE JR., 1985)
2. Order-Crossover (OX1): (DAVIS, 1985)
3. Order-Crossover (OX2): (SYSWERDA, 1991)
4. Position-Based Crossover (POS): (SYSWERDA, 1991)
5. Heuristic Crossover (HX): (GREFENSTETTE, 1987)
6. Edge-Recombination Crossover (ER): (WHITLEY; STARKWEATHER; FUQUAY, 1989)
7. Alternating-Edge Crossover (AE): (GOLDBERG; LINGLE JR., 1985)

2.3.1.3 Mutation Operator

Combined with the selection method and the crossover operator, the mutation operator is another tool used by genetic algorithms to accelerate convergence and avoid local optimum solutions. Implementing a mutation operation after crossover and selection grants the algorithm the capability of generating entirely new, and possibly better, offspring in situations in which the solutions would otherwise stabilize at a local optimum. This happens in situations in which the parents are the same or too similar, which tends to be more likely to occur in the late generations near the end of the execution. Multiple types of mutation operator have been described in the literature, their concept is always focused on creating small disturbances on the chromosomes without distancing them too much from the original sequence. Among the multiple algorithms some worth mentioning are:

1. Simple Swap or Twors Mutation: An operator that simply swaps two genes on the tour, swapping their predecessors and successors.

2. Reverse Sequence Mutation (RSM): In the reverse sequence mutation operator, a sequence S is taken from a given chromosome and the genes in this sequence are copied and then inversely placed on the sequence again.

3. Throas Mutation and Thrors Mutation: Two similar operators in which three random selected genes are swapped according to a defined rule (this rule is the difference between Throas and Thrors methods).

4. Partial Shuffle Mutation (PSM): This operator performs a shuffle operation in a sequence S taken from a tour, scrambling the order of the genes on this sequence.

2.3.1.4 Selection Methods

Selection methods are another important part of the problem since it's their responsibility to choose which of the individuals in the current generation will be used to reproduce offspring aiming towards a next generation with higher fitness. The selection operator must be carefully formulated to ensure that the best members of the population (with higher fitness) have a greater probability of being selected for mating, but also ensuring that worse members will still have some probability of being selected, avoid convergence to the nearest local optimum. Almost all selection strategies studied in the literature are based on a survival of the fittest mindset, being the most common described below (RAZALI; GERAGHTY, 2011):

1. Tournament Selection: In his kind selection, some individuals of the population are selected randomly and compete against each other, the individual with the highest fitness wins and is selected as a parent for the next generations. This method has the advantages of being very simple to implement and not requiring sorting, scaling well as the number of cities increase and allowing a parallel implementation.

2. Proportional Roulette Wheel: on this selection strategy individuals are selected with a probability that is directly proportional to their fitness values. The probabilities of selecting an individual can be seen as spinning a roulette wheel (thus the name) with the size of the segment for each individual being proportional to its fitness. This strategy has a downside when it comes to convergence since, depending on the fitness distribution, the solution may converge to fast towards a local optimum, or too slow towards any point, both undesired behaviors.

3. Rank Based Roulette Wheel Selection: is a strategy where the probability of an individual in the roulette wheel is based on its fitness rank relative to the entire population. The rank-based selection method first sorts individuals according to their fitness and then calculates selection probabilities according to their ranks rather than fitness values. This method is considered one of the best methods for selection in GA because it is able to maintain a constant and adjustable pressure in the evolutionary search, controlling convergence. The downside of this method is, however, that it requires a sorting algorithm to be put in place, which can become time-consuming as the number of cities rise.

2.3.2 Simulated Annealing

Simulated Annealing (SA) is a heuristic optimization method based on the idea that the process of finding (near) optimal solutions for optimization problems is similar to the physical annealing process used to obtain low energy states on metals. The main

concept behind a Simulated Annealing optimization algorithm is that the optimization search starts on a high energy state, accepting different candidate solutions whether they are near-optimal, optimal or not good ones. As the process runs, however, the search algorithm starts to cool down, reducing its willingness to accept bad candidate solutions and focusing on the good ones.

Being proposed in 1983(KIRKPATRICK; GELATT; VECCHI, 1983), Simulated Annealing is, thus, a stochastic method used to avoid local, non-global optimal results to be considered as optimal ones. This is done by accepting, based on a heuristic probability function, worst solutions that would otherwise be ignored and that may lead to optimal solutions once the local minimal point is transposed. However, during the process the chance of accepting these deteriorations is decreased, meaning that after a certain point the algorithm accepts that it has transposed local minimum and is getting closer to the global minimum, accepting only solutions that are better than the current on from this point on.

As for the process, the algorithm starts by generating a random or quasi-random candidate solution and performing local searches to optimize this initial solution iteratively. On initialization, after generating the first candidate solution, the algorithm links it to an initial temperature T_0, which represents the degree of refinement of such solution, or its internal energy state. On each iteration, a given number n of local searches are performed on the candidate solution and accepted or not based on the acceptance probability for the current iteration, after that the current temperature T_i of the optimization process on iteration i is decreased. In case one of the local searches find a solution which is better than the current candidate, this improvement is automatically accepted and the improved solution becomes the current candidate. Otherwise, if the local search finds a solution x that is worse than the current candidate this solution is tested against a probability P_x of being accepted according to its evaluation $f(x)$ and the current temperature T_i as shown in Equation 7.

$$P_x = exp\left(-\frac{f(i) - f(x)}{T_i}\right) \tag{7}$$

The $f(i)$ on the term $f(i) - f(x)$ on Equation 7 stands for the evaluation of the current candidate, meaning that the probability of a worst solution x being accepted depends on how worse than the current solution it is.

The convergence of this type of algorithms to find near-optimal solutions for optimization problems has widely been proved in the literature, some works, as (DEKKERS; AARTS, 1991), provide an extensive mathematical analysis of simulated annealing convergence. Such a convergent behavior is closely related to the decreasing exponential formulation of the probability P_x which decreases the chance of a poor solution being accepted exponentially as the temperature decreases, granting that at the end, the algorithm is just accepting the best solutions it finds. The initial temperature parameter is similarly important for the convergence once, if too low, can lead the algorithm to never accept a solution, and, on the other hand, if too high, leads the algorithm towards accepting lots of solutions, delaying convergence.

A graphical representation of the workings of a Simulated Annealing solution as described above can be seen in Figure 3

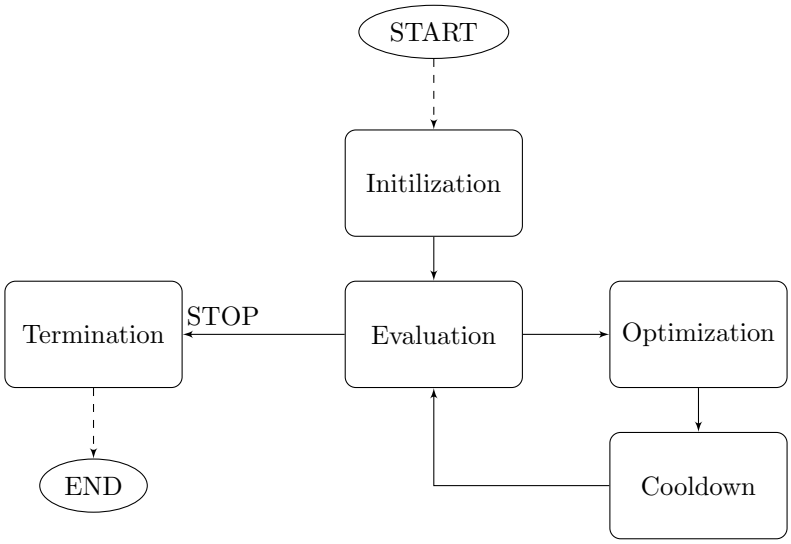


Figure 3: SA Fluxogram

Source: Prepared by the author

Similarly to genetic algorithms, simulated annealing (SA) approaches were firstly intended to solve generalized combinatorial problems, needing adaptations to correctly handle TSPs. The adaptations simulated annealing are, however, much simpler that the ones required by a Genetic Algorithm.

The process to use simulated annealing to solve the TSP problem does not differ much from the general case, the representation of the solution and the local search algorithm, however, must be correctly selected. As for the representation, a simple list of cities organized on the order they will be visited on the candidate solution tour is a simple, usual, and straight forward method representation method. The local search algorithm, or neighborhood selection move, however, presents a more complicated choice. Many different algorithms and moves have been reported in the literature, including even genetic algorithms (XU; LI; GUO, 2017) being used as SA local search methods in more sophisticated approaches. The most common local search moves used on SA algorithms applied to the TSP problem are (ZHAN et al., 2016), (GHANDESHTANI et al., 2010):

1. Switching, also called Swapping: Randomly selects two nodes from the tour and exchange, or swap their positions each other. In general, four edges between cities will be replaced by this operator.

2. Translation: a move that randomly selects a sequence, or a portion of the tour, i.e. the cities between two cities i and j, and inserts it somewhere else, creating a neighbor tour in the process.

3. Inversion: a move that randomly selects two cities i and j inverse the cities between positions i and j, replacing two edges, or paths between cities, in the process.

4. 2-opt: An algorithm that randomly removes two edges from the tour and reconnects the two paths created by this operation. Basically the algorithm breaks the tour into 2 parts then reconnects the 2 parts in the other possible way, not reinserting the same edges it removed from the tour again, but creating new ones in the process.

5. 3-opt: Works just like the 2-opt, but instead of removing two edges at once, it removes three edges, separating the tour in 3, and reconnecting it in another fashion afterward.

6. k-opt: uses the same idea of 2 and 3-opt, separating the tour in k parts and reconnecting it again. It is important to notice that both 2 and 3-opt are special cases of the k-opt move, however, for any k>3 the move takes more computational time and does not provide a significant improvement over 2- and 3-opt.

7. LK (Lin-Kernighan): LKH (LIN; KERNIGHAN, 1973) is a kind of $variable$-Opt heuristic that tries to choose the most suitable value of k at each iteration step.

2.3.3 Ant Colony Optimization

Ant Colony Optimization (ACO) (DORIGO; BIRATTARI; STUTZLE, 2006; DORIGO; CARO, 1999) is another biologically inspired approach to solve optimization problems. The paradigm behind this kind of solution is inspired by the communication between social insects, ants in this case, that cooperate with each other to reach their objective in nature. Because of their high cooperative societies, ants have inspired a great number of algorithms, most of them based on their communication mechanisms, based on the deposition of pheromones in the environment. In nature, ants communicate through the deposition of chemical markers on the ground (pheromones), the idea used by ACO, distributing virtual ants to perform searches on a set of candidate solutions and to indicate, through virtual pheromones, the way to find better solutions.

Following the natural inspiration, ACO is an iterative optimization algorithm in which a number of virtual ants build a set of candidate solutions to the problem, and communicate the quality of these solutions through the deposition of virtual pheromones. Interpreting the trails of pheromones, virtual ants of the next iteration have guidelines on how to build better candidate solutions and iteratively refine the search space until an optimal or quasi-optimal solution is found.

As a simple and very useful algorithm, ACO has been used in many fields and adapted to solve many different applications. Some of the most common variations of such algorithms are the Ant System (AS) (DORIGO; MANIEZZO; COLORNI, 1996) and the Max-Min AS (STÜTZLE; HOOS, 2000), both widely used for optimization purposes.

Figure 4 presents a graphical representation of the algorithm described above.

As in the other methods, ACO also needed some modifications to handle the TSP. Besides GAs, ACO algorithms are among the most studied for TSP usage, therefore many applications and variations have been created to better approach the problem. The approach behind the usage of ACO in TSP, is distributing the ants in the cities and let them walk around trying to complete a tour as if they were on the nature walking from their nest to food sources, and back. At the end of this exploration process the ants deposit pheromones on the trails they took between the cities to perform their tours. The intensity of the pheromone an ant will deposit is directly related to the length of the tour they performed, the shorter the tour, the strongest the pheromone. On the next exploration round, the ants will attribute to the trails, or edges between cities, a probability of being followed that is directly proportional to the amount of pheromone on that edge. This system ensures that every edge has a chance to be followed, thus avoiding local minimal once, but creating a trend towards better solutions on every iteration. This common behavior to the solutions is graphically represented on the diagram of Figure 5.

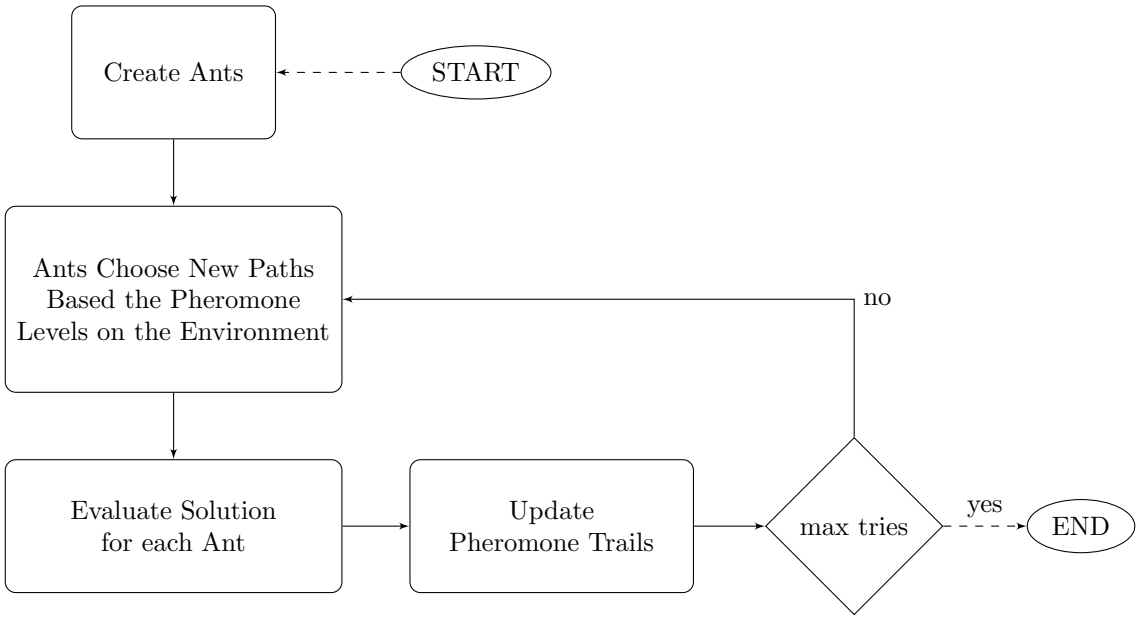


Figure 4: ACO Fluxogram

Source: Prepared by the author

Besides this common approach, the different algorithms (STÜTZLE; HOOS, 2000; TUBA; JOVANOVIC; JOVANOVIC, 2013; CHENG; MAO, 2007) differ on how they attribute pheromones to the edges, on how they evaluate new solutions, and on how they create these solutions. Many algorithms also perform a mid-algorithm optimization, performing local search iterations on candidate solutions after an ACO iteration is concluded, optimizing the ants' paths, thus the pheromone trails, before a new generation of ants is let lose on the cities (SU; HLAING; KHINE, 2011). Some of the most notable adaptations of ACO are presented below:

- Ant System (AS): AS was the first ACO applied to the Traveling Salesman Problem (TSP). Initially, AS was proposed in three different versions:ant-density; ant-quantity; and ant-cycle. While in ant-density and ant-quantity the ants updated the pheromone trail only after moving between two cities, in ant-cycle the pheromone trail update was done only after all the ants had constructed their tours, causing the pheromone trail to be a function of the tour quality. Among the three versions, ant-cycle was experimentally proved to provide the best results, outperforming both ant-cycle and ant-quantity. These results, however, despite promising for small TSP problems, proved that AS does not scale well with larger instances of TSP.

- Ant Colony System (ACS): ACS (DORIGO; MANIEZZO; COLORNI, 1996; DORIGO; CARO, 1999) introduces some adaptations that are supposed to improve this algorithm's performance when compared to AS. In ACS, for example, the pheromone is added only to arcs belonging to the global-best solution, which for large TSP instances provides a better result than deposing pheromones on the iteration-best solution arcs. ACS also implements a mechanism aimed to make an already chosen arc less desirable for the next ants. This mechanism improves the the exploration of not yet visited arcs, maximizing the solution potential, and avoiding local optimal

solutions. These adaptations, along with a more aggressive probability function ruling the choice of new paths being chosen, provide a significant increase on the quality of the solutions when compared to AS, at least for large TSPs.

- $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS): $\mathcal{MM}$AS (STÜTZLE; HOOS, 2000). introduces the idea of limiting the strengths of pheromones trails. In this algorithm, the intensity of pheromones trails is bounded between a minimum and a maximum predefined values. This technique has the ability to to avoid search stagnation, keeping the solutions away from local optimum solutions. $\mathcal{MM}$AS also, as in ACS, only one ant, usually the iteration or the global best, is allowed to deposit pheromones after an iteration, accelerating the convergences towards better solutions.

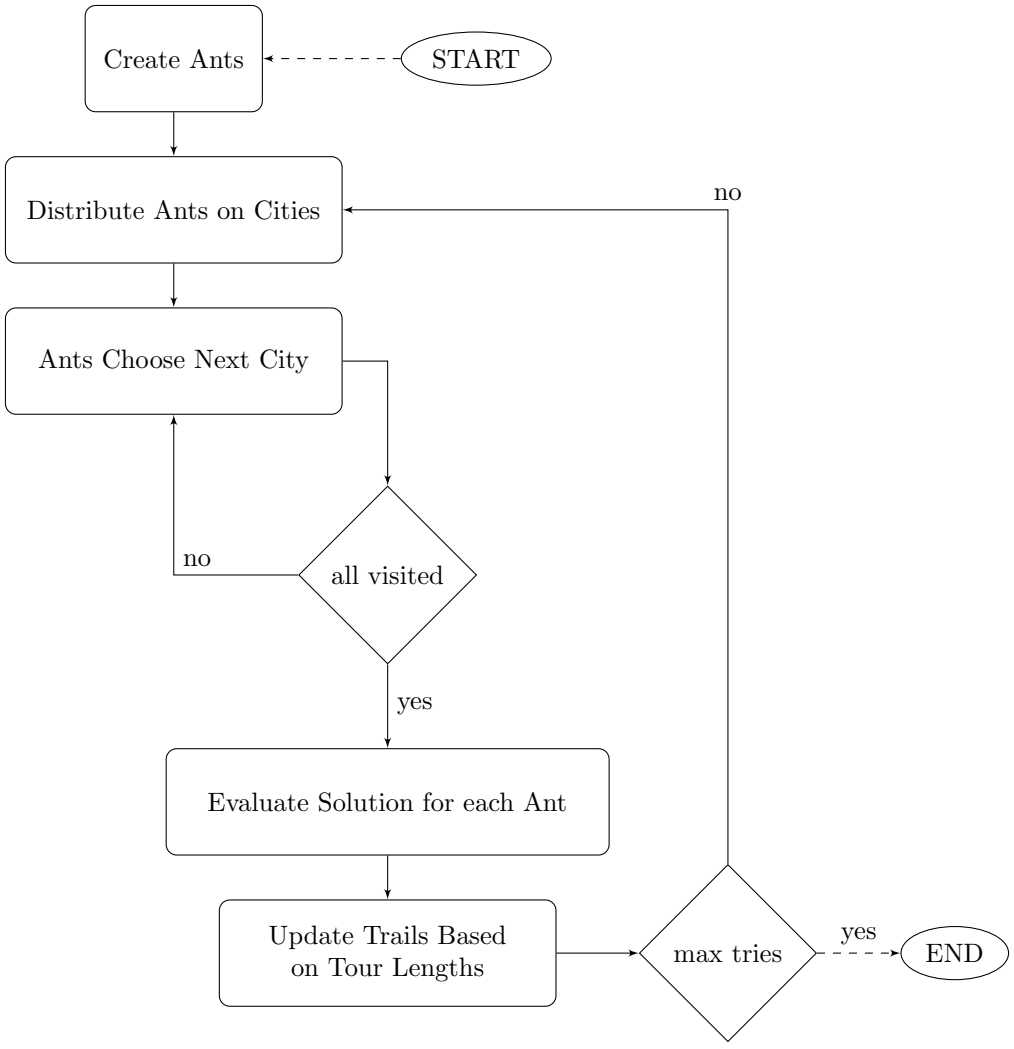


Figure 5: ACO for TSP Fluxogram

Source: Prepared by the author

2.4 Auction Theory and The Assignment Problem

In their book about auction theory Menezes and Monteiro (2005) define an auction as a mechanism used by a market to balance supply and demand. According to their interpretation auctions are mostly used to sell goods or resources that do not have a solid or established market and have the important characteristic of having well-known rules for price formation. These characteristics make auctions, in their words, "more flexible than a fixed price sale and perhaps less time-consuming than negotiating a price" (MENEZES; MONTEIRO, 2005, p. 10).

Auctions can also be explained as settings in which players have to choose their strategies in a limited information context. According to this definition, an auction can be seen as resource allocation mechanism with two main rules: An allocation rule, specifying who gets the object after the auction; and a payment rule, describing how much the winner must pay for this object.

Complementing these interpretations, an auction algorithm is an intuitive method for solving the Classical Assignment Problem. This problem has fundamental importance to multiple practical contexts, economics, game theory, production planning, networking and even transportation problems can be translated into some form of the Classical Assignment Problem, which can be adapted from (BERTSEKAS, 1992) as follows:

Definition 2 (The Classical Assignment Problem)**.** *Given a group of n agents $a_1, a_2, ..., a_n$ and another group of m resources $r_1, r_2, ..., r_m$ that have to be matched on a one-to-one basis, and a benefit function B_i^j for matching resource j to agent i, the objective is to find the best assignment between agents and resources, i.e. pairs (a_i, r_j), to maximize the total benefit of the system. In other words:*

$$\underset{B_i^j}{\textit{maximize}} \sum_{i,j} B_i^j \tag{8}$$

These kinds of problems, can also be looked at from an economic equilibrium perspective (BERTSEKAS, 1992). Under these optics, the goal of an auction, or assignment, is to find a economic equilibrium state such as that every buyer is satisfied with the price it payed for a given good. Supposing that an object o_j has a price p_j, and that the person who receives it must pay p_j. Then, the value of o_j for people i is $V_i^j = a_i^j - p_j$, where a_i^j is the benefit function defined in Def. 2. Logically, each person wants to buy, or be assigned, to an object that maximizes the perceived value V_i^j, or, in other words:

$$a_i^j - p_j = \max_{j=1,...,n} \{a_i^j - p_j\} \tag{9}$$

When (9) is satisfied for all n buyers, the system is said to be in $equilibrium$, providing maximum total benefit, and solving the assignment problem successfully.

Auctions can have many sets of rules considering how bidders behave in different types of auctions. The underlying assumption about auctions is that each buyer has an intrinsic value for an item being auctioned, and that he is willing to purchase the item for a price up to this value, but not for a higher price. Considering this behaviour, four main different types of auctions can be defined (EASLEY; KLEINBERG, 2010):

1. Ascending-bid auctions, also called, English Auctions, are those open bid auctions in which the bidders increase their bids in turns until no bidder is willing to raise

their bid above the current highest bid. Usually, bidders drop out until finally only one bidder remains, and that bidder wins the object at this final price.

2. Descending-bid auctions, or Dutch auctions, are open descending price auction. This kind of auction is known for its use in selling flowers in the Netherlands. Bidding starts at a high price that continuously decreases until someone accepts the current price and pays it.

3. First-price sealed-bid auctions: Each bidder bids having no knowledge about the price offered by the other bidders, the winner is the contestant who offered the highest bid.

4. Second-price sealed-bid auctions, or Vickrey auctions (VICKREY, 1961), are those in which each buyer submits a sealed bid simultaneously. In this kind of auction, however, the highest bidder pays the value of the second-highest bid.

Assignment problems and auctions are much used on Multi-agent Systems, or MAS. These systems can be defined as a network or group of agents that work together to perform a task or solve a problem that, for its nature or characteristics, cannot be performed by any agent alone. Each agent is, therefore, responsible for individual and specific tasks that are part of the overall solution. These attributed tasks can be performed with moderately efficient results depending on the agent skills, characteristics or internal state. So it becomes necessary for these tasks to allocated as optimized and conveniently as possible, which is an assignment problem.

One of the most know and used protocols that solve the assignment problem in MAS is the CNP proposed by Smith in (SMITH, 1980). The CNP (Contract Net Protocol) is based on a market economy paradigm and implements a mechanism of contract expediting and follow-up, specifying the interaction between agents for fully automated competitive negotiation through the use of contracts. A great number of extensions and adaptations were presented to modify the CNP since 1983. Some of them, like the ones proposed by (WALSH; WELLMAN, 1998) and by (GERKEY; MATARIC, 2002), modify the algorithm to make it work in a distributed way, while others introduce parallelism and fault tolerance, like the one proposed by (AKNINE; PINSON; SHAKUN, 2004).

2.5 Human Mobility Models

The study of human mobility models is an important asset in several practical contexts. For urban planning purposes, for example, it is important to know how pedestrians behave to plan the placement of streets, sidewalks and crossings, as to not perturb the pedestrian flow in crowded cities. These models are also important for architects and engineers involved on the project of subway tunnels, airports, and other indoor spaces that may be subject to dense pedestrian activity. Communication scientists and network engineers, on the other hand, are interested on these models because they can be used to model mobile network usage statistics, and antenna placement requirements for cellular and wireless networks.

According to the literature, pedestrians tend to take the fastest route their destination (KRETZ, 2009), regardless of whether this route is crowded or not (HELBING et al., 2001). Each person has an ideally preferred walking speed, which depends on several factors, such as age, sex, and constitution. The average comfortable walking speed, however, usually varies between $1.2m/s$ and $1.6m/s$ (COSTA, 2010).

As one can expect, crowd density and pedestrian flow affect walking speed and direction. (FROHNWIESER et al., 2013) conducts a study on how these statistics relate to each other. The authors conclude that people tend to change their speed more often as crowd density increases, a phenomenon that probably occurs when pedestrians are trying to avoid others by walking faster to escape the crowd, or slowing down to let crowd pass. This work also finds that people change their walking direction to greater degrees in high density situations, probably trying to dodge other people in the way, causing their speed to momentously decrease.

(FANG; LO; LU, 2003), complements this motion study modelling the movement of each individual in terms of the walker self motivation and the interaction between himself and the people around it. In this model, an individual is subject to three forces which act upon its movement speed: the forward interaction force, which is exerted by the people in front of or behind the walker; the lateral interaction force, which is exerted by the people walking left or right the walker; and the individual self-driving motion force, or their self self motivation to escape the crowd. The first force represents the will of the pedestrian to keep a distance from the person walking in front, stopping or slowing down to avoid making physical contact. The second force is exerted by people to avoid jamming with other people from the sides, slowing down in order to avoid collision. The third force, represents the desire of an individual to escape from the control of the crowd by overtaking the people in front.

It is important to notice that, as corroborated by (FROHNWIESER et al., 2013) the formulation presented by (FANG; LO; LU, 2003) is highly dependent on the distance between pedestrians, or, in other words, the crowd density. (NELSON; MACLENNAN, 1995) presents the concept of an optimal crowd density of 1.9 $persons/m^2$, at which, at least theoretically and under the best conditions, a given crowd could reach its maximum flow rate. Even though crowd densities can reach high numbers such as 4, or even 7, $persons/m^2$, in most applications such high numbers are not often verified.

3 RELATED WORKS

Coordinating mobile robots to monitor moving targets, as proposed in this work, is no simple task. For once, the coordination strategy depends on factors such as communication, which can be a challenge in dynamic environments with moving targets and limited resources, like hardware size and power or battery charge. In addition to that, there is also the aspect of whether the coordination should be centralized, decentralized or distributed, inserting or not points of failure on the system and making it, or not, capable of taking self-healing actions. The mobility or maneuverability of the robots can also be an issue, while large UAVs or UGVs present mobility restrictions that may not be desirable depending on the application, light commercial-off-the-shelf (COTS) UAVs, such as quad-copters, do not present such restrictions, but have a more limited operating range. Hardware or resources support is also an important factor, COTS UAVs present an affordable solution for smart-surveillance applications, but cannot support power hungry hardware, limiting the communication range or the image sensing capabilities of such assets, a limitation that is not always observed on UGVs or fixed-wing UAVs, which may support complex, heavy and power demanding hardware resources. Because of these different characteristics the usage of UAVs, UGVs or robotic-based systems to monitor and observe multiple moving targets has been vastly researched. While some researches, like in (ADAMEY; OZGUNER, 2012), focus their solution is spacial clustering, dividing the observed area targets into sectors and assigning each robot to a sector, other approaches, such as proactive assistance demanding, which can was introduced in (KOLLING; CARPIN, 2007), are also found in specific literature.

Many of these different approaches are analyzed in (KHAN; RINNER; CAVALLARO, 2018), which organizes, categorizes and compares 20 years of studies in the usage of mobile robots for observing multiple moving targets. This work lists and addresses five factors, which are common in this area, and characterizes this kind of problem, providing a classification method to evaluate the different approaches. The authors further group the existing approaches based on four major control techniques: Cooperative Tracking (CT) (KIM; CRASSIDIS, 2010; ADAMEY; OZGUNER, 2012; OH et al., 2015), which has the objective to persistently track moving targets; Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) (PARKER; EMMONS, 1997), which aims to increase the collective time of observation for all targets; Cooperative Search, Acquisition, and Track (CSAT), which alternates between search and track states, both searching and tracking moving targets; and Multi-robot Pursuit Evasion (MPE) (CAI et al., 2008), whose objective is to capture evasive targets.

Following the classification proposed by (KHAN; RINNER; CAVALLARO, 2018) Cooperative Tracking (CT) approaches are those focused on reducing the time duration between two observations to the same target, updating the information the system has on

the location of the targets on a more frequent basis. CT is very similar to the approach presented here in this current thesis. This work, however, does not only implements CT, but also a distributed algorithm to do so, which does not require a master, or central, node to work. This fact reduces the chances of miss-functions due to single points of failure and communication problems, the same strategy used in (ELMOGY; KARRAY, 2008; SUJIT; BEARD, 2007; CAI et al., 2008; YADAV; MENG, 2014). In addition, a comparison regarding the type of targets considered can also be made. According to the classification presented in (KHAN; RINNER; CAVALLARO, 2018), targets here are considered non-cooperative, as in (ELMOGY; KARRAY, 2008), which means that they do not broadcast their positions, as in (PACK et al., 2009), but also does not necessarily mean they are hostile, or evasive, as (CAI et al., 2008), which is the case in the problem handled here.

An example of a proposal exploring a type of CT approach is presented in (KIM; CRASSIDIS, 2010), in which a centralized algorithm is used to maximize the visibility of ground targets in urban environments. The algorithm proposed by the authors groups the targets to maximize each UAV's coverage area in order to reduce overlap. Then it assigns each UAV to a circular optimal path that maximizes the visibility given the shapes and the locations of obstacles on the ground. Their scenario considers that the number of moving targets to be tracked is much larger than the number of aircraft. However, they consider that the aircraft fly much higher than a common multi-rotor small UAV, being more suitable to be deployed on large fixed wing aircraft than on cheap COTS UAVs.

The same kind of approach is used in (ADAMEY; OZGUNER, 2012). In this work, a clustering algorithm is used to define an area of surveillance for each UAV, making them responsible only for the targets existing in the area they were assigned for. The authors, however, use in their work an interesting data fusion algorithm to estimate each target's position, giving a different insight into movement prediction problems and techniques. Another similar technique is used in (OH et al., 2015), in which an offline algorithm calculates a terrain division and allocates each UAV to a given set of targets present on one of these divisions. Then, the UAVs assume an orbital movement behavior around these places. This kind of solution works against the main constraints of the problem defined in this thesis, the need to have a group of mobile and unconstrained agents capable of moving around according to the possible changes in the operation scenario.

Besides Cooperative Tracking, another technique described in (KHAN; RINNER; CAVALLARO, 2018) is the Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT), which has the goal to dynamically position robots to maximize the collective time during which targets are observed. CMOMMT differs from CT since, instead of trying to keep track of the targets positions through successive visits, i.e. occasionally leaving the target and later returning back, it tries to maximize the time a target effectively stays into the field of view of the UAV. The problem addressed by CMOMMT is to maximize not only the number of targets under observation, but also the duration of observation for each target, which is a much more complex problem. A variation of such algorithm is presented in (KOLLING; CARPIN, 2007), where the authors implement a "help" mode, in which a robot losing a target from its FOV, broadcasts a help request to the other robots. The robots available then respond to these help requests by approaching the robot that is in need of help, maximizing the usage of resources, and granting that no target is lost.

Target tracking is also addressed in (CHEN; CHANG; AGATE, 2013) where the authors proposed a dynamic UAV path-planning algorithm for tracking a ground target.

Their algorithm is a combination of a point-mass approximation and other techniques, such as vector fields and obstacle avoidance strategies. These techniques give the proposed method the ability to find the shortest path for an UAV tracking a target that may be moving in an environment that may include obstacles and/or wind.

Approaching specifically crowd control and pedestrian monitoring, (BURKERT; FRAUNDORFER, 2013) brings an important study in pedestrian groups behavior and image processing to evaluate potentially threatening behaviors. The work of these authors can be considered an important complement to the solution presented here in this present thesis since there is no focus on image processing here. An additional module would be needed to be run on each UAV to identify targets and make sure they are found by the UAV cameras. This aspect is out of the scope of this work thus considered transparent here.

While still on the same topic, (ABELEDO et al., 2013) distances itself from the UAV or aerial monitoring problem and discusses definitions and challenges to address TDTSP situations, presenting a mathematical and a complexity analysis, along with possible algorithms to solve such problems. (HELVIG; ROBINS; ZELIKOVSKY, 2003),continues discussing this type of problem presenting a special case of the TDTSP in which not only the cost to move between cities change over time, but the cities' (comparable to the targets in the present work) locations also change. In their work, the authors also present suitable techniques to solve this problem, as well as an intense discussion and analysis of these type of problems.

Closer to the work here presented, (ENGLOT; SAHAI; COHEN, 2013) provides an analysis of both greedy and LKH heuristics (LIN; KERNIGHAN, 1973) to solve TSP problems focused on the pursuit of moving targets. This work has a very similar goal to the one here presented and many of the considerations presented in their work were also used to conceive the solution reported in this current thesis. Their solution, however, is much less flexible than the one presented here, not allowing the mobility and temporal constraints required by pedestrian tracking problems.

Even tough Cooperative Tracking techniques share a common goal with the proposal here presented, which is reducing the interval between two observations of a same given target, they considerably differ from the work here proposed. For once, most of these algorithms and solutions focus themselves in spatial clustering or target grouping, allocating each UAV to a specific portion of terrain or group of targets and, mostly, forcing them into orbital or circular flight paths. This kind of approach may be a good fit to larger fixed-wing UAVs, which are designed to be used on higher altitudes, with reduced maneuverability, and covering large portions of ground through the usage of complex and heavy image acquisition equipment. However, it might not be the best approach once a small, cheap and lightweight UAV that flies closer to the targets, covering smaller portions of ground, but possibly in urban environments, is considered.

Table 1 presents a synthesis of some of the most important multi-target tracking and observation works discussed in this chapter, highlighting the techniques they used and the problem specific constraints they were designed to deal with.

Table 1: Multi-target Observation Related Works

Work	Technique	Problem Specifics
(KIM; CRASSIDIS, 2010)	Target grouping and clustering; Observation moving in circular optimal paths	Presence of Hostile Targets that try to avoid observation by either change their speed or hiding behind obstacles
(ADAMEY; OZGUNER, 2012)	Two Level Approach; Region Clustering; Individual path planning within the assigned region	
(OH et al., 2015)	Vector field guidance control; Observation moving in standoff orbit; target clustering/assignment	Standoff tracking of moving target groups considering the mobility restrictions of fixed-wing UAVs
(KOLLING; CARPIN, 2007)	Movement vector with improved target loss prediction; implementation of "help" mode, which calls for help when losing a target is unavoidable	Problem considers evasive targets in an open region
(CHEN; CHANG; AGATE, 2013)	Tangent-plus-Lyapunov Vector Field Guidance;	Capable of tracking targets moving randomly, or according to way-points, i.e. roads in complex environments that may include obstacles and/or winds.
(SUJIT; BEARD, 2007)	Target grouping and assignment through a distributed sequential auction system	Targets are bound to an enclosed area and no $a\ priori$ information on the location of the targets is known.
(ELMOGY; KARRAY, 2008)	Target Grouping Approach; Extended Kohonen neural network to track groups of targets; Assignment of target using an auction algorithm to reduce the number of active robots	Considers two different parts of the problem: the single target tracking problem, and the multi-robot cooperation problem.

Source: Prepared by the author

4 PROBLEM FORMULATION

4.1 Problem Outline

This work focuses on the development of a distributed and autonomous control system for UAVs to be used in crowd control and monitoring scenarios to support law enforcement operations. In this context, this work assumes crowd monitoring as an operation of threat or target identification and position tracking. That assumption means that the goal of the monitoring application is to track and record the position of the potential threats. However, this does not mean that the system continuously keeps recording their every single move and act, but it keeps track of their approximate location, and periodically returns to that location to verify their behavior. In terms of the system's behavior, the UAVs must distribute targets among themselves and keep track of these targets without losing them under any circumstance. This also means that the UAVs responsible for more than one target must be able to periodically visit each target to assure it is still where it ought or was predicted to be, recording his actions and behavior

This multiple target observation problem can be classified as a Cooperative Tracking (CT) problem. This class of problems is focused on reducing the time duration between two observations to the same target, updating the information the system has on the location of the targets on a more frequent basis. Particularly, this problem can also be described as a combination of two common problems, an assignment problem, and an MT-TSP-like problem, in which, given groups of targets and UAVs, the targets have to be assigned to the UAVs such as to maximize the performance of the TSP problem. Considering this information, the problem can be defined as follows:

Definition 3 (Problem Definition)**.** *Given a set of n targets $T = \{t_1, ..., t_n\}$ each one moving independently at a velocity $v_t i$, the state of target t_i at time t can be defined in terms of its position (x,y and z coordinates) as follows:*

$$\chi_i^t = (x_i, y_i, z_i)^t \tag{10}$$

and a set of m pursuer UAVs $U = \{u_1, ..., u_m\}$ on the same environment, moving with velocity v_u, where:

$$v_{uj} \gg v_{ti} \qquad \forall t_i \in T, \quad \forall u_j \in U \tag{11}$$

The observation state of target t_i at time t by UAV u_j, which has a sensor with FOV F_j, is defined as follows:

$$O_{ij}^t = \begin{cases} 1; & \textit{if} \quad \chi_i^t \in F_j^t \\ 0; & \textit{otherwise} \end{cases} \tag{12}$$

The problem is to find the best assignment between UAVs and targets, i.e. pairs (u_j, t_i)*, that maximizes the observation of all the targets during the mission, or, mathematically:*

$$\underset{O_i}{\textit{maximize}} \sum_{i}^{n} \sum_{j}^{m} O_{ij}^t \tag{13}$$

and plan the motion of every $u_j \in U$ *such as to find a tour that intercepts all targets paired with* u_j *and minimizes the average time between consecutive distinct observations to the targets.*

4.2 Design Assumption

The defined problem, besides restricted in terms of its objective, can become extremely generic if some assumptions concerning the targets, the operating environment, the UAVs, and the observation premises are not made. The next sections approach each of these important factors, presenting assumptions aimed to create a realistic scenario that ensures the system usability in real life situations, defining a solvable problem and avoiding situations in which the problem can become too hard or unsolvable.

4.2.1 Target Model

As the approached problem can become prohibitively complex, some assumptions are made about the dynamic behavior of the targets to create a controlled scenario closest as possible to a real-world situation in which this monitoring system could be employed. It is supposed in the application scenario that targets belonging a mob infiltrated in a demonstration displaced on a map, will have a well defined movement behavior, not always keeping the same speed of movement, but having a controlled variability in their walking patterns. This means that they will never go from full-stop to full-sprint in a matter of seconds, for example. The application scenario also assumes that the targets do not abruptly change their direction of movement, acting, therefore, as if they were walking in a street or avenue.

These assumptions are very likely to be true in real life scenarios since people in a demonstration are not likely to move too fast nor to sprint due to the agglomeration of people. Demonstrations also tend to follow a well-defined path, with well defined directions, normally following streets or avenues in a city, and not running around at will, reducing, therefore, the movement direction variability. Thus, by considering these assumptions, this work bases itself on a well defined set of rules concerning the targets that ensure it's usability in real life situations, defining a solvable problem. In case this assumptions were not true, as in situations in which targets move too fast for the UAVs or in which their movement cannot be predicted, the addressed problem could become unsolvable, meaning that, in some cases, the UAVs would not be able to monitor targets all satisfactorily.

Another assumption made in this scenario is that the UAVs are going to be used to monitor pacific crowds, with the number of potential threats limited to a few individuals

on the crowd. This can also be the case considering, for instance, lone wolf terrorist attacks. This system is not intended to operate on extreme situations where hundreds of threatening individuals are jointly operating, riots or violent manifestations are examples. The system can be used in such cases, but the success in monitoring great numbers of targets will not be guaranteed since it depends heavily on each target location and speed.

It is also considered that targets are non-cooperative, which means that they do not broadcast their positions or information about themselves. It also means that targets are not necessarily hostile, or evasive, trying to escape observation, which is also the case on most demonstration or human gatherings.

4.2.2 Environment Model

As explained, the system is intended to be used mostly in urban environments, or at least in environments where human gatherings are more likely to occur, such as streets, avenues, parks, entertainment venues, sports stadiums and beaches, for example. All of these environments are well structured and have a common set of constraints that can be used to characterize such interest areas. Among these characteristics, there are two of utmost importance to this work, which are assumed as to create controlled and tractable scenarios: steadiness and boundeness.

The first characteristic is related to the non-evolution of the environment. It is assumed that environments do not evolve, meaning that their limits, or bounds, do not change and that no obstacles to the UAVs or to the sensors, or cameras, appear during the missions.

The second characteristics means that the environment is well-bound, in other words, the environments have clear and fixed boundaries, or limits, that cannot be trespassed by the targets. This assumption is very likely to be true in real life since people cannot just walk through buildings, fences or walls.

4.2.3 UAV Model

It is very important to be noticed that this system is aimed to be deployed in light commercial-off-the-shelf (COTS) UAVs, such as quad-copters, which present an affordable solution for smart-surveillance applications, but cannot support power hungry hardware, limiting the communication range and capabilities of such assets. It is also assumed that, since lightweight UAVs cannot support complex and heavy image acquisition equipment either, they are more suitable to be used closer to the targets, covering smaller portions of ground.

However, COTS UAVs compensate for not supporting demanding hardware with their maneuverability. Small UAVs can be maneuvered in any kind of environment, even indoors, as in convention centers or sports gymnasiums, making them a perfect choice to monitor targets from a close observation point, or in situations where maneuverability is necessary.

4.2.4 Coordination and Observation Premises

It is also assumed here that this work is mainly focused on the functional behavior of the UAVs, leaving aside problems like movement control and image processing, which can be very complex on their fields, but that are out of the scope of this work and then not addressed here. In the context of this work, it is supposed that UAVs are capable of recognizing the targets once they are on sight of the camera or image acquisition device mounted on them. Thus, the image acquisition and processing is transparent to the

handled problem. Communication control is also left out of the scope, assuming that a network controller module is able to maintain a reliable communication network between the UAVs on the system, and forward the messages as necessary. It is also assumed that the low-level control of the actuators are outside of the scope of this work

In order to provide the reader a comprehensive summary of the scope considered on this work, Figure 6 illustrates each of the main SW and HW modules existing on a single drone and how they interact with each other. The specific context here is represented on the figure by the grey box entitled `Crowd Monitoring Mission Controller`, which communicates with the movement controller, the image processing modules, and the communication controller as needed to perform the observation mission.

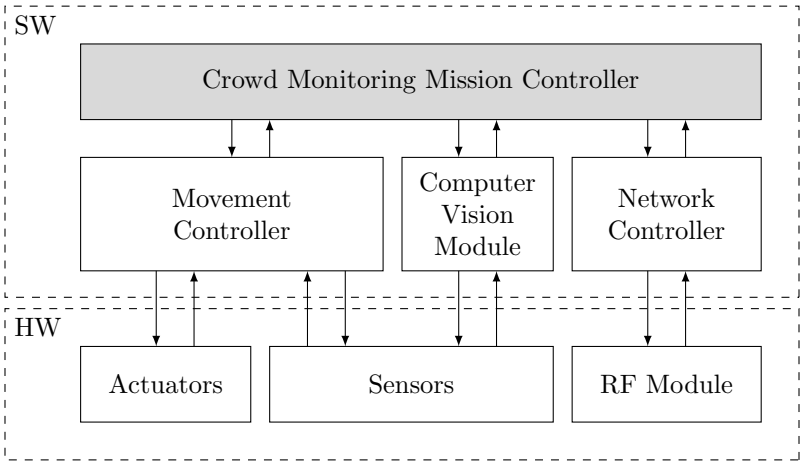


Figure 6: Solution Context and its Interface with Other Modules

Source: Prepared by the author

5 PROPOSED SOLUTION AND IMPLEMENTATION

The conceived solution is composed of 4 main modules or algorithms:

1. Auction Algorithm: used to distribute threats, or targets, among the UAVs taking into account the current status of both UAVs and the targets, such as speed, location and the set of targets each UAV has already assigned to itself

2. Optimization Solver: responsible for defining which of the targets in the UAV's current queue it will visit next. Moreover, it also organizes the next visits to not lose sight of any of the targets the UAV is responsible for.

3. Handover Mechanism: similar to the "help" technique proposed in (KOLLING; CARPIN, 2007), granting that if a UAV finds out it will not be able to cover all its targets, it informs and negotiates with another UAV to take the responsibility for one or more targets, handing it(them) over and reducing its current target queue

4. Movement Predictor Method: uses numeric methods to predict where a UAV should go to intercept a target, calculating the target's next position based on its observed ground speed.

These algorithms are presented in detail in following sections, where their inner workings and how they interact to build the complete solution are discussed. In order to provide the reader a comprehensive flowchart, the overall system is represented in Figure 7, where the cited modules are numbered accordingly to enumeration above.

5.1 Solution Overview

The integrated solution uses 4 main modules, one providing an auction mechanism, another implementing a tour solver, one responsible for the target handover method, and the last predicting the targets movements for better interception and observation. These modules interact with each other to make sure all the mission requirements will be met, i.e. the modules interact and exchange information in order to make sure all targets will be monitored and will not be lost. Together these different algorithms work toward the same goal: ensuring that a given target is visited by some UAV every n seconds. The variable n here stands for the maximum time interval allowed between two visits to the same target without giving it time to disappear from the field of view of the system as a whole. The solution works to ensure that there will always be an UAV capable of monitoring or checking on a given target before the visit time window of that specific target expires.

To achieve this desired result, the first step of the proposed solution is the distribution of targets among the UAVs. Every target known by the system must be, at first, assigned

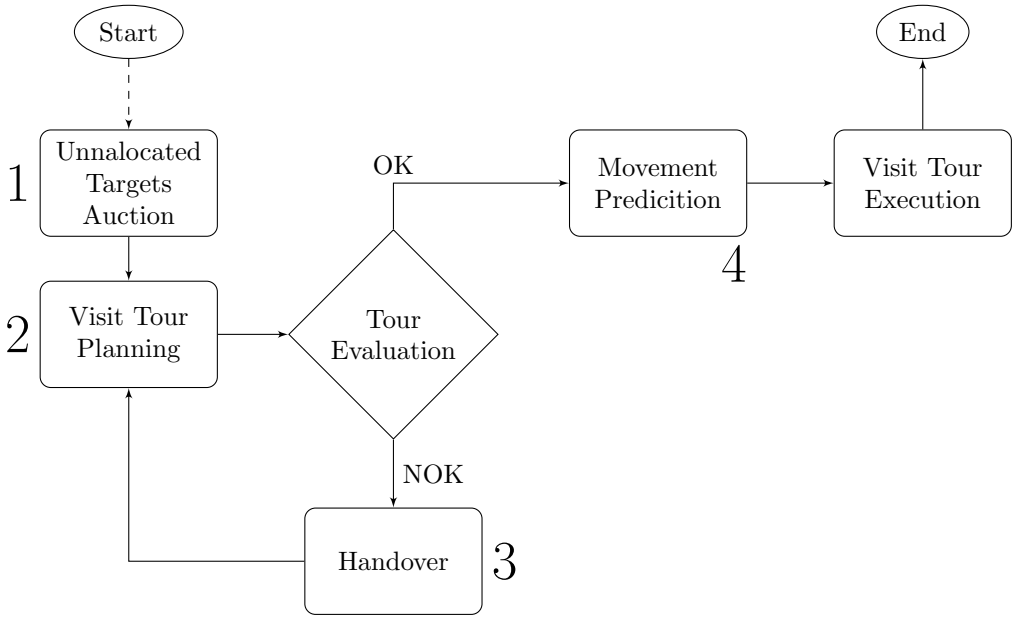


Figure 7: Proposed Solution Flowchart

Source: Prepared by the author

to the UAV that fits better to monitor it at that given time. This allocation is made through an auction algorithm specifically tuned to the desired result of this system. Each auction section allocates a target to an UAV taking into account the state of both UAVs and targets, in terms of their location and speed, and the number of targets already assigned to each UAV. The Auction mechanism is asynchronous to the rest of the system. It is activated only when either a new target is discovered or informed to the system, or when a handover operation has to take place.

Independently from the auction sessions, a tour solver runs internally on the UAVs that have at least one target assigned to them. The purpose of this optimization solver is to calculate and find the best order to visit the targets an UAV has assigned to it. This is executed to make sure they will not be lost. To do this, each UAV simulates some possible different orders to visit the different targets. Then, it analyses each of these different orders and chooses the best tour (order), to visit the targets. It is important to notice that this algorithm is conceived to, and it tries to, minimize the UAV's displacement between targets in two successive visits. This minimization makes possible multiple visits to a faster UAV while only one to a slower UAV, if necessary (e.g. in systems in which there are a combination of UAVs with different capabilities and that can move with different maximum speeds). If the solver finds a case in which it predicts that the best route or solution it finds will not ensure that the maximum time between visits to a target will be respected, it triggers the handover mechanism.

The handover mechanism removes a target from an UAV current queue and reallocates the responsibility for this target to another UAV. By doing this, it is ensured that an UAV having problems to visit all its targets will be relieved from at least one target. Then, it recalculates its route, trying to visit all remaining targets within their visit time constraints. Basically, when the handover is triggered by the genetic solver, the UAV tries to sell one of its targets, the one located in a more distant position from the other targets it has to visit.

Once an auction is triggered by one UAV, another one, that fits better to the auctioned target, will buy it. Once an UAV succeeds selling a target, it will try to re-execute its tour solver, recalculating its new route for the remaining target. If a feasible solution is not found, this handover process may be repeated until a solution is found, or until the UAV that triggered the handover keeps just one target under its responsibility.

After distributing targets, calculating the best tours, and handling targets over, the last step on the whole process is predicting where a target will be next so that it can be assertively observed by the UAV. This role is fulfilled by the movement predictor, that considers the current mission time and the last speed observed for a given target to calculate where it should be intercepted by the UAV responsible for it.

A detailed flowchart of the algorithm comprising the whole solution can be seen in Figure 8 for a better understanding of these mechanisms and how they interact The reader should notice that, in this figure, the modules of the system are also numbered as they were in Figure 7, detailing how and where each of the methods presented is inserted on the system.

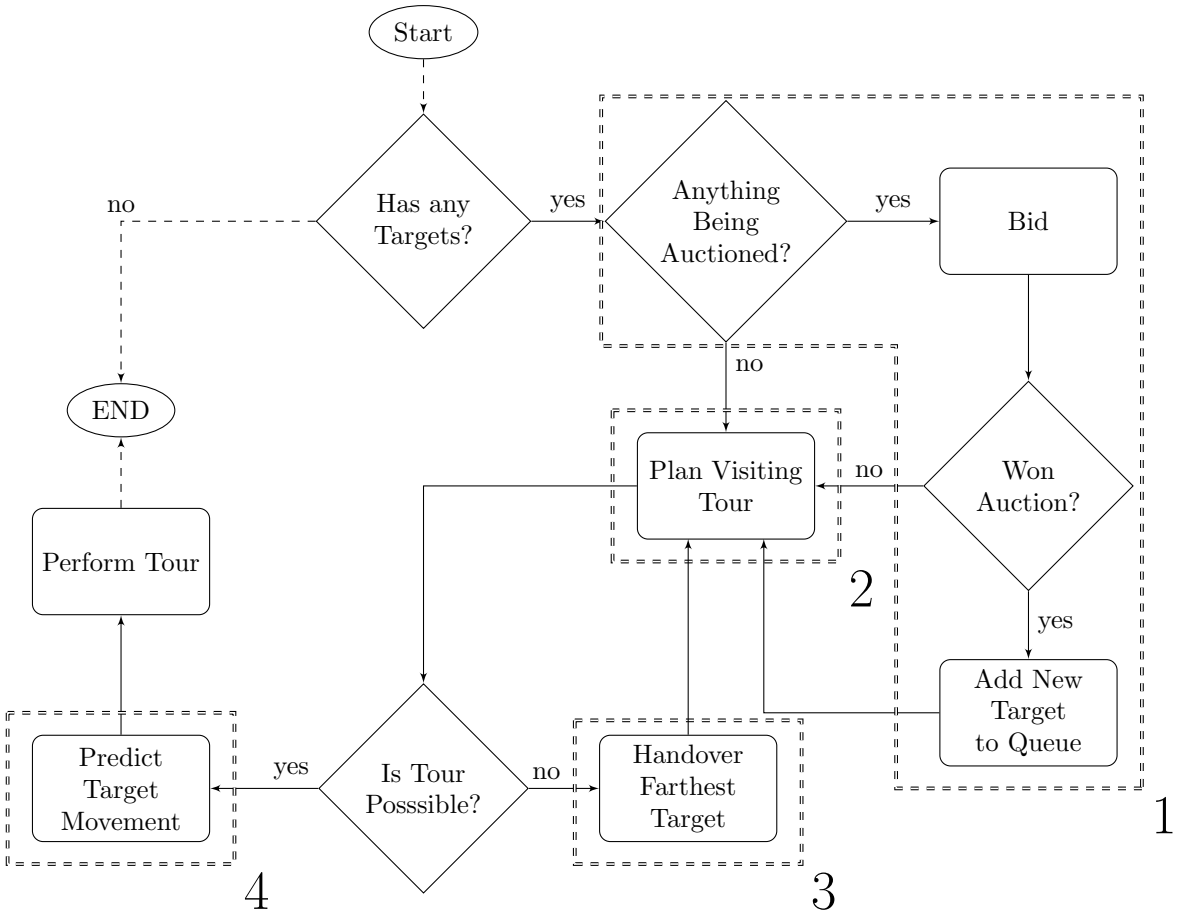


Figure 8: Detailed Flowchart of the Proposed Solution

Source: Prepared by the author

5.2 Auction Mechanism

As mentioned, an auction mechanism was proposed to be used as a task allocator, allowing an efficient distribution of target monitoring responsibilities among the UAVs. The idea behind the algorithm is based on the behavior of buyers and sellers of an auction market, meaning that goods or resources will be put to auction and sold to the buyer who makes the best offer for them. In the presented context, the targets represent the goods being sold and each UAV represents a buyer that bids, or not, for a target it considers it is able to monitor efficiently.

The implemented mechanism works through a first price sealed-bid auction based on a simplified auction protocol. In this kind of approach, the decision of who wins or not the auction is made based on the value of the bids. These bids must be well calculated and formed as to ensure that, as wished, only the fittest bidder will win. Bids are calculated following a set of rules that define how each parameter of interest to the auction will influence on the bid. To solve the problem of distributing the moving targets among UAVs, a mathematical rule was created based on both UAVs' and targets' current state. (14) shows how this rule was implemented.

$$Bid(u,t) = K_a \left(\frac{1}{v_t(D_{ut} + K_d \sum_i^{n_{uq}} D_{ti}) n_{uq}} \right) \tag{14}$$

Basically, as is shown in (14), each bid of an UAV u for a target t is inversely proportional to the number n_{uq} of targets u has queued for monitoring. This ensures that UAVs that already have targets leave new targets to other UAVs, not overloading themselves. The value of a bid is also inversely proportional to the velocity v_t of the target t, so UAVs bid less for fast targets that will potentially be a problem to be monitored. The other parameters in this rule are the distances between the UAV and the target (D_{ut}) and the distance (D_{ti}) between the target being auctioned and the other n_{uq} targets already assigned to the UAV. These two parameters ensure that an UAV will bid higher for targets that are close to each other and close to the UAV itself. The rationale for this way of functioning is distributing targets spatially close to each other to the same UAV. As a consequence, not only an UAV will have to move less to monitor its assigned targets, but also free UAVs will tend to win the rights over targets that are not close to any other target. Therefore, the goal of this implementation is, mainly, forcing UAVs to choose to monitor groups of targets that are near to each other, bidding less for isolated targets. In order to adjust the effects of the variables on the BID the constants K_a=1m^2/s and K_d=25 were empirically chosen accordingly, through experimental analysis, to guarantee that the these desired effects are achieved. While K_d is a dimensionless factor responsible for adjusting the effects of D_{ti} on the overall result, K_a is a constant used to balance both n_{uq} and v_t effects.

Each of the auction sections in this work is composed basically of five main phases. In the first phase, called Announcement, an auctioneer agent advertises all possible bidders about the starting of a new auction session. During the second phase, the auctioneer waits for bids for a certain amount of time before proceeding to the next phase. On the third phase, bids are analyzed, the auctioneer chooses the winner, and the bidders are informed the result. During the next phase, the contract is expedited to the winner, giving it the right to use the auctioned resources. In the last phase, the auctioneer waits for every bidder, including the winner, to acknowledge that the auction has ended. The phases

where communication is necessary the system implements a 3-time retry mechanism. This mechanism ensures that errors or non-responses on a section due to communication issues are less prone to occur. This process is graphically represented in Figure 9.

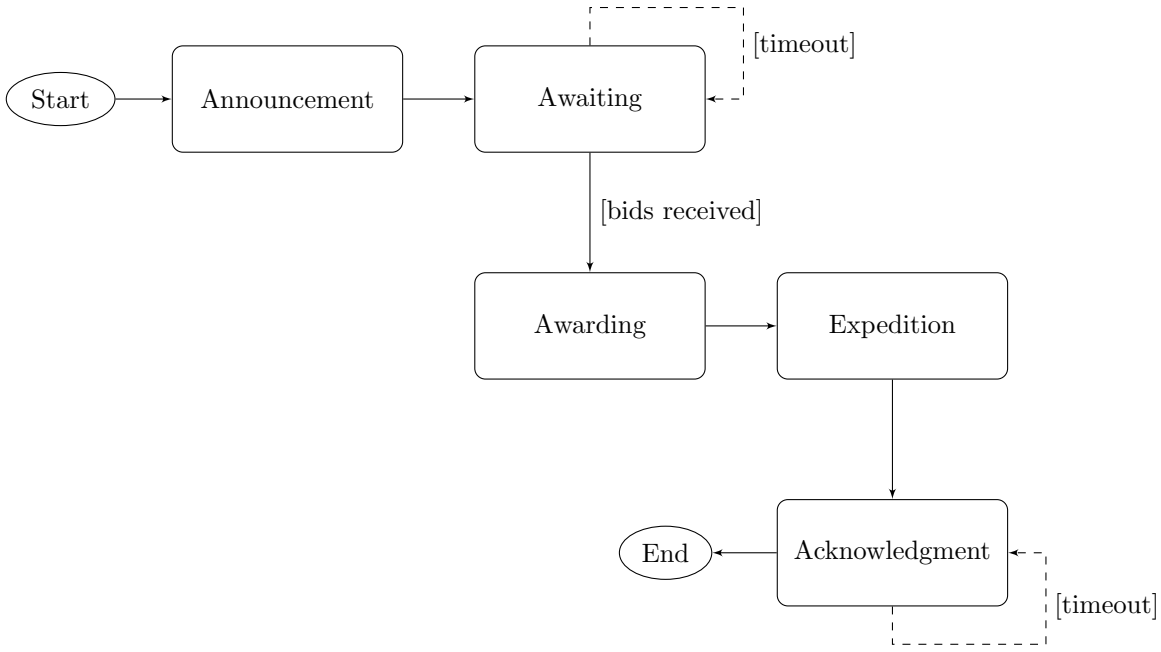


Figure 9: Auction Mechanism

Source: Prepared by the author

Additionally, since the auction and observation environments are both highly decentralized, there is no node in the system with knowledge over the location of the UAVs and the targets to be used on the auction sections. In order to provide this information for the bidding UAVs, the auctioneer, which is the entity (UAV or system user) auctioning a target, passes the most recent information about the target being auctioned in the moment it announces the auction. It is also important to notice that, for auctioning purposes, the information the auctioneer provides about the target is considered accurate, with no noise or other kind of artifact being taken into account by the UAVs when they bid.

5.3 Handover Mechanism

The Handover algorithm is simply a target assignment transference mechanism. The implemented solution is activated once an UAV finds out, by analyzing the solution output provided by the genetic solver, that it will not be able to visit all targets on the next tour without losing one or more of them. Once this fault is detected and the mechanism activated, the UAV chooses the target that is farthest from all the others. This is done by calculating the mean distance between each target to all the others. Then, the UAV starts the auction for this target, granting that some other UAV will bid and assign itself to handle this target.

The auction mechanism used here is the same auction mechanism used in the first allocation phase. The only difference is that now, the UAV auctioning the target is not allowed to bid for it since it would make no sense for this UAV to try buying something it

already owns. As shown in Figure 8, it is important to notice that the handover algorithm may be run more than once, removing as many targets as necessary from an UAV responsibility and stopping only when a viable solution is found by the GA or when the UAV has only one target assigned to it.

During development it was observed in preliminary results that certain targets tended to be handed over continuously, with some targets reaching the mark of two thousand handover in 15 minutes (900s), more than one handover per second. Further investigation showed that this behaviour appears when a UAV hands over a target to another UAV, which by its turn, immediately also decides to hand over the same target, and this continues indefinitely back and forth. In order to avoid a difficult target to be passed around infinitely, once an UAV takes the responsibility for a given target, it has to keep this for at least ten seconds. The UAV may consider skipping visiting this target, i.e. the UAV may not handle it during this time interval, but it is not authorized to pass it over before this timeout. This behavior is usually enough to change the system status, affecting bids and tours, thus avoiding this possible infinite handover situation, just by the natural movement of the UAVs and the targets, reducing the number of handovers from thousands to a few dozens during a 15 min simulation. Also, in every handover all the available UAVs always bid for a target, meaning that there will always exist a UAV to take over the target being handed-over.

As stated above, the conditions for activating this mechanism are related to the solution output provided by the GA solver, which is supposed to be the optimal, or quasi-optimal, solution for the set of the targets it was run over. The trigger is associated to a threshold on the solution fitness number, if the fitness of the GA solution is higher than this threshold level, meaning the average maximum distance targets move on that turn is greater than the threshold distance, the handover method is called. The threshold value of 20 meters was chosen according to the field of view of a RaspBerry PI camera, model Sony IMX219PQ[5] mounted on an UAV flying at an altitude of 45m. The camera features an 8Megapixel 3.674 x 2.760mm CMOS sensor and uses 3.04mm lenses, resulting on a field of view of 62.2 x 48.8 degrees and on a ground blueprint of 54.326 x 40.855 meters, or a circle of 20 meters centered on the image. Table 2 presents different ground blueprints for the camera field of view for an UAV flying at different altitudes.

Table 2: Ground Blueprint of a Sony IMX219PQ[5] Camera Module Mounted on an UAV Flying at Different Altitudes

Altitude (m)	10	20	30	40	50
X Footprint (m)	12.0	24.1	36.2	48.2	60.3
Y Footprint (m)	9	18.1	27.2	36.3	45.3

Source: Prepared by the author

5.4 Optimization Algorithm

Since an UAV on the monitoring system may be responsible for several targets, it must be able to decide in which order it will visit these targets. The visits must be organized in a way to make sure that no target can move enough to be lost between two consecutive

visits. This means that, by having visited a target A and proceeded to visit others targets, an UAV must return to visit A again before this target is able to move out of this UAV's field of view. The problem presented by this challenge is similar to a Moving Target Travelling Salesman Problem (MT-TSP).

Exact solutions for TSP problems are hard and very resource consuming to find. In order to obtain the optimal solution to a TSP, it would be needed to analyze all possible different routes and choose the best one. However exact, this kind of brute force approach is not suitable for real-world applications, presenting a computational complexity cost of $O(n!)$. They quickly become too time-consuming to run on commercial machines or embedded processors, even for small TSP problems. Because of these high costs to find the optimal solution, many different AI, Optimization and Heuristic methods are found in the literature to find acceptable local optimal solutions to specific cases of the problem.

On this context, the goal to be achieved by the optimization algorithm on this work is the organization of the visits in a way to make sure that no target can move enough to be lost between two consecutive visits. The optimization problem here is the minimization of the average maximum displacement between visits to the same target, considering all targets under the responsibility of a given UAV. Mathematically, this objective function can be expressed as follows:

$$min(average(f_A, f_B, f_C, f_D, ...)) \tag{15}$$

,where f_X $(f_A, f_B, f_C, f_D, ...)$ is the maximum distance target X (A,B,C,D...) moves between any two visits to it's position. In other words f_X can be calculated as:

$$f_X = max(d_1, d_2, d_3, ...) \tag{16}$$

,where d_1 is the distance that target X moves between the first to the second visit, d_2 the distance X moved between the second and third visits, and so on.

To solve this special problem, an optimization module was specifically implemented to deal with these requirements. Basically, each potential solution to the problem is represented by an array of ordered targets that represents the visitation order of that specific solution. For instance, a solution represented by the genome ACD will guide the UAV to visit first the target A, then the target C and finally target D. Figure 10 shows a sample tour and the graphic representation of the solution it represents. The numbers close to the arrows provide information about the order each node of the graph is visited. On the implemented methods, each tour is twice as long as the number of targets the UAV running the algorithm has assigned to itself. This means that if an UAV has n targets in its monitoring queue, the tour of the possible solution to that UAV specific TDSP problem will have size $2n$. The double-length was implemented so that faster targets may be visited more frequently than slower ones in the same tour, making it easy to monitor them minimizing the risk of they disappear. According to this last definition, each target may be visited twice on the same solution or visiting tour. However, this is also possible to happen that some targets are visited multiple times while others are only visited once.

Each of these potential solutions is analyzed according to the fitness function presented in (15) and (16). In the example presented in Figure 10, target A is visited three times on the tour. Supposing that between these visits A moved d_1 meters between the last visit and the first visit in this tour, d_2 meters between the first and the second visit,

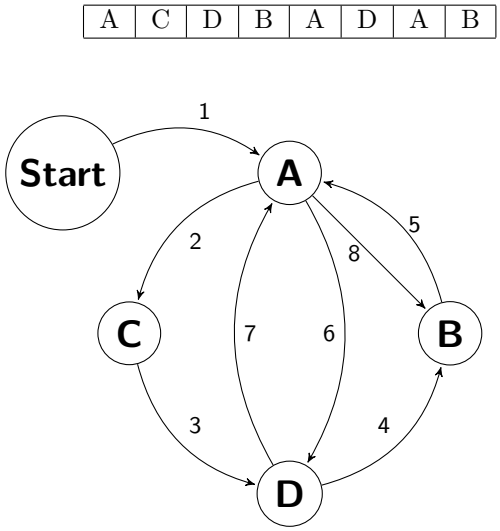


Figure 10: Visit Diagram Representing a Possible Optimization Solution with Tour ACD-BADAB

Source: Prepared by the author

and d_3 meters between the second and third visit, the maximum value $f_A = max(d_1, d_2, d_3)$ will be chosen as the fitness of A in this solution. The fitness F_{sol} of that particular solution is then calculated as $F_{sol} = average(f_A, f_B, f_C, f_D)$.

In order to evaluate the best heuristic method to solve the problem at hand, three of the most common different heuristic and artificial intelligence methods were implemented, and their performances to solve the proposed MT-TSP investigated. These three methods, their implementation, and the modifications applied to solve the problem are discussed on the following subsections, while a comparative analysis of their performance can be found on Section 6.3 .

5.4.1 The Solution Using Ant Colony Optimization

The implemented ACO method is based on the $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) approach (STÜTZLE; HOOS, 2000). This method has the advantages of converging faster towards an optimal solution and exploring faster the search space in pursuit of a global optimal. At the end of each generation on the $\mathcal{MM}$AS, only the best ant marks the environment with pheromones, a remarkable difference from other algorithms in which all ants mark their trails. Besides, on this approach, the amount of pheromones deposited on each edge between targets is bound on the upper side by a max value, and on the lower side by a min value. The intensity M of the pheromone an ant deposits on edge e in time t is, thus, calculated in terms of its tour fitness F and the upper max and lower min bounds as described in (17):

$$M_e(t) = \left[M_e(t-1)(1-\alpha)) + \frac{1}{F} \right]_{min}^{max} \tag{17}$$

where α represents a temporal decaying constant and the operator $[x]_{min}^{max}$ means that $M_e(t)$ will be bound assuming the min or max values in case it reaches these thresh-

olds.

Following the idea presented in the diagram of Figure 5, each ant is distributed on a given target, or city, and chooses their next city based on the pheromone level of the edges leading to the cities it has not visited yet, according to the following probability function:

$$P_i^j(t) = \frac{M_{ij}(t)\frac{1}{d_{ij}}}{\sum_0^n M_{in}(t)\frac{1}{d_{in}}} \tag{18}$$

meaning that the probability of an ant choosing to go to city j from city i among n cities is directly proportional to the pheromone level M_{ij} on the edge that links i to j, and inversely proportional to the distance d_{ij} between these cities.

5.4.2 The Solution Using Simulated Annealing Optimization

The simulated annealing algorithm implemented for this TSP problem is quite straightforward. At the beginning a candidate tour is randomly generated, and, as the simulation goes, a local search algorithm is used to generate a new solution. This new solution may or may not be accepted according to the criteria previously presented in both Figures 3 and (7).

The local search method chosen for this implementation was a simple 2-opt method, a method that takes two edges between targets and swaps their extremities, keeping the tour closed, but still generating a neighbor candidate solution similar to the initial one. The main idea behind a 2-opt, or any k-opt local search algorithm, is to take a route that crosses over itself and reorder it so that the crossing disappears, potentiality improving the solution.

5.4.3 The Solution Using Genetic Algorithm Optimization

The genetic algorithm implemented to solve the TSP problem on this work is based on a Path Representation(POTVIN, 1996) of the tours, over which a Rank Based Roulette Wheel Selection(RAZALI; GERAGHTY, 2011) is first applied, followed by both PMX Crossover and RSM Mutation operators.

When it comes to representing solutions on genetic algorithms, path representation (POTVIN, 1996) is considered to be the most classical and natural representation, representing tours by a list of the cities ordered as they appear on the tour. The crossover operators based on this representation typically generate offspring that inherit the relative order of the cities. This representation, however, has to be checked for uniqueness, since each tour can be represented in $2n$ distinct ways because any city can be placed at the first position.

A Rank-Based Roulette Wheel Selection(RAZALI; GERAGHTY, 2011) is a strategy in which the probability of an individual in the roulette wheel is based on its fitness rank relative to the entire population. The rank-based selection method first sorts individuals according to their fitness and then calculates selection probabilities according to their ranks rather than fitness values. This method is considered one of the best methods for selection in GA, because it is able to maintain a constant and adjustable pressure in the evolutionary search, controlling convergence. The downside of this method is, however, that it requires a sorting algorithm to be put in place, which can become time-consuming as the number of cities increases. As the algorithm developed on this work is intended to be used in real-life-like problems with a limited number of cities, the used of a Rank Based solution is justified once the sorting will not be too time-consuming.

In the Reverse Sequence Mutation operator, a random sequence S is taken from a tour and the genes in this sequence are copied and then inversely placed on the sequence again. Besides simple, the RSM mutation operator has shown promising results in terms of convergence when compared to other methods(ABDOUN; ABOUCHABAKA; TAJANI, 2012).

In its turn, the PMX crossover (GOLDBERG; LINGLE JR., 1985) is a crossover operator method created to perform one point crossovers on Path Representations. Taking two parents $p1$ and $p2$, PMX constructs offspring by changing the positions of the cities on $p1$ one by one, with the cities on $p2$ that occupy these same positions. However, in order to keep the solution valid, the cities in these positions are not just overwritten, they are swapped. This way, to set position i to city c, the city in position i and city c swap positions, granting that all cities are on the tour, and that no city is repeated unnintentionally.

5.5 Numeric Movement Prediction Module

The movement prediction module works by taking as input a list of way-points containing the initial positions $\vec{s_0}$ of the targets in time t_0 according to the tour order of the solution under evaluation. Following from t_0 and $\vec{s_0}$ the method applies a numeric approach to to find the interception position s_i of the next target by the interceptor. The process is repeated for each target on the way-point list always calculating the interception point of the next target on the current tour or mission time, and departing from the current UAV position.

The whole modules is based on the idea that, for a given UAV moving with speed $\vec{v_u}$, and a target moving from point $\vec{s_t}$ with speed $\vec{v_u}$, the interception point will be the one in which the UAV, after moving for a period τ, finds itself on the same point as the target. Mathematically, considering the position of the target as the origin, this situation can be written as:

$$\vec{v_u}\tau = \vec{v_t}\tau + \vec{s_t} \tag{19}$$

ultimately, applying scalar product in both sides as to eliminate the vectorial properties, Equation (19) can be rewritten as:

$$(\vec{v_u}\tau) \cdot (\vec{v_u}\tau) = (\vec{v_t}\tau + \vec{s_t}) \cdot (\vec{v_t}\tau + \vec{s_t}) \tag{20}$$

which is the same as:

$$|\vec{v_u}|^2\tau^2 = |\vec{v_t}|^2\tau^2 + |\vec{s_t}|^2 + 2(\vec{s_t} \cdot \vec{v_t})\tau \tag{21}$$

reorganizing:

$$0 = (|\vec{v_t}|^2 - |\vec{v_u}|^2)\tau^2 + |\vec{s_t}|^2 + 2(\vec{s_t} \cdot \vec{v_t})\tau \tag{22}$$

Solving equation 22 for τ and using this value in equation 19, the point of intersection between the UAV and the target can be easily found. The algorithm presented on this work makes use of the Secant Method (CHAPRA; CANALE, 1998) to find the roots of Equation 22, a quicker solution than using square-root calculation methods, which are very resource consuming.

6 EXPERIMENTAL RESULTS

6.1 Simulation Set-up

In order to correctly evaluate the proposed solution and analyze its behavior as realistically as possible, a simulated environment capable of modeling the behavior of both UAVs and targets was needed. Such a simulator needed to be able to mimic the real time constraints of this type of system, considering targets' and UAVs' characteristics and ensuring that the processing platform would be able to handle all the concurrent tasks involved in the simulations. To achieve this goal, a custom ad-hoc simulator was developed. Since the environment and scenarios proposed in this work involve more than just the UAV movement, most simple commercial simulators were not fitted to be used in the evaluation, or did not provide the required flexibility for such analysis , thus the need for a custom ad-hoc solution .

In the proposed evaluation scenarios, when working with high-speed vehicles and moving targets, for example, it is important to assure that no outside variables influence the simulation results. At high speeds and strict temporal constraints, even the slightest delay in processing some information or making some decision can heavily influence the activity outcome, sometimes crossing the line between success and failure. When monitoring threatening individuals on a crowd, for example, the system must be able to process a mission and target related information as soon as they come, treating them as a priority, risking to lose targets otherwise. With these constraints in mind, a real time environment approach had to be developed to ensure that no outside variables would influence the simulation results.

To solve this simulation problem and simulate real time behavior, an Ubuntu Linux kernel was used together with the real time interface RTAI and the Jamaica Java virtual machine. The RTAI interface is an extension to the Linux kernel that features a hardware abstraction module, patching the kernel and improving process scheduling, letting users develop applications with rigid timing constraints under Linux. Furthermore, when programming real time applications under Java, garbage collectors become a problem because they preempt running applications to organize the virtual machine heap. To avoid preemption from influencing on the system, JamaicaVM was used since it is a hard real time oriented Java VM that features a preemptable, deterministic garbage collector. The approach ensures that garbage collectors will work only when the system is idle and will not interrupt important processing.

Besides considering the real time constraints of the application, the system was also supposed to deal with the high level of concurrency presented by the simulated scenarios. At a same moment during a simulation, both UAVs and targets are supposed to be moving. UAVs are supposed to be exchanging information over a communication network and

they are also supposed to run their own control loops and internal algorithms. To achieve such concurrency, each UAV instance was modeled as a completely separated thread, acting, thus, independently from all other threads, sharing with them just the simulation environment parameters through a communication channel. Besides the UAV mission controller instances, three other modules were also implemented as threads to maintain concurrency: an UAV movement engine, responsible for managing UAVs movement behavior; A target movement engine, responsible for managing target's movements; and a Simulation Control Module, responsible for controlling the simulation parameters and simulate the communication network, besides supporting all other modules. While these three last modules effectively simulate the physcal and logical behaviour of the system, it is important to notice that UAV mission controller instances are not really part of the simulator since their objective is to effectively run the coordination system, and not simulate it. For better understanding, the four types of threads and the general multithreaded architecture are diagrammed on Figure 11. Since the system is supposed to mimic real time constraints and concurrency, the simulation must be run on a computational platform capable of running all threads simultaneously, running any two of them sequentially would break the concurrency principle, breaking the very concurrency concept.

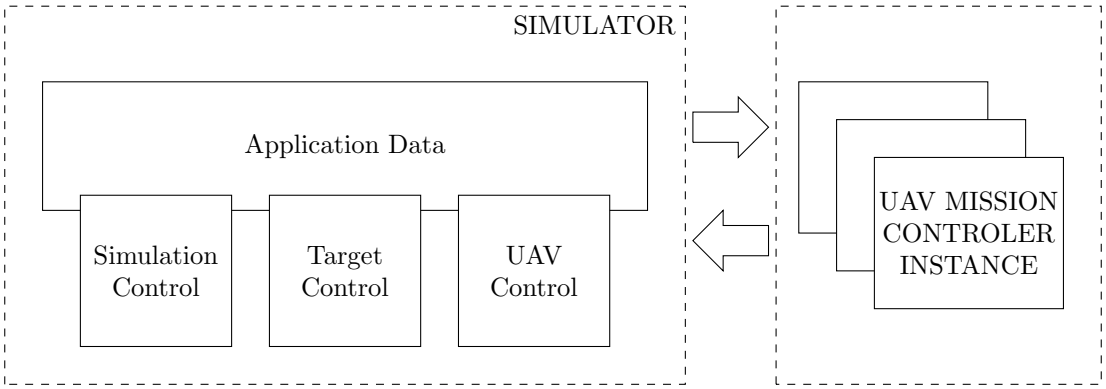


Figure 11: Threaded Simulation Environment

Source: Prepared by the author

The simulator and the experiments were also parametrized accordingly to mimic the behavior of the 3DR Iris+ quadcopter (3DROBOTICS, 2014), a mini-UAV manufactured by 3D Robotics. The Iris+ 950 kV motors are capable of accelerating the 1282g UAV to around 20m/s, the flight autonomy stands between 16 and 22 minutes, depending on the payload. Considering this information, and the assumptions made on Section II of this work, the initial test environment was parametrized as presented in Table 3. Target speeds were based on common movement speeds of individuals walking on a crowd.

The experiments presented in Section 6.2 were all run using the Genetic Algorithm as heuristic to solve the TD-TSP problem. The choice for the GA was due to its better behavioral and temporal performance to solve the proposed problem, as presented in Section 6.3. In order to address the overhead created by the execution of a complex heuristic method on a embedded platform, the genetic algorithm stop conditions were set to allow it evaluate only 125 possible solutions. The choice for 125 evaluations, consisting on 5 generations of 25 individuals each, was made based on the observed performance of the algorithm during multiple experimental runs, in which it was observed that this value

Table 3: Simulation Parameters

Simulation Time	900s
Simulation Area	500x500
Number of UAVs	5
Number of Targets	25
Target Initial Speed	random from 0 to 2m/s
Target Initial Heading	random
Target Speed Variation	random from $\pm$0m/s to $\pm$0.5m/s changing each 20s
UAV Speed	20m/s
Heuristic Stop Condition	125 evaluations (5 generation of 25 individuals)

Source: Prepared by the author

provided near-optimal results for TD-TSP instances with up 8 or 9 targets. This choice, besides optimized for larger numbers of targets, presents a poor performance in cases with only 2 or 3 targets. In these cases, a brute force approach would find the optimal result using fewer comparisons, demanding less processing power. In order to minimize the impact of this tradeoff, the choice of wether use an heuristic or a brute force approach for few targets could be left to the user of the system. In this evaluation, however, it was considered that all cases use the heuristic approach.

6.2 Simulation Results and Analysis

6.2.1 Correct Behavior Tests

To properly evaluate the developed solution, a set of initial tests were designed to test if the system has a correct behavior according to what was expected, using the parameters presented in Table 3. With this goal in mind, a set of 6 simulated scenarios was created. In the first four scenarios, the objective was to test the solution comparing 4 different cases with part of the solution modules active or deactivated. The first scenario is a simple run of the algorithm with no handover operation turned on and also no prediction on the targets' movement, meaning that at each visit the UAV searches a target on it's last known position, where it was last seen. The second simulation scenario has the handover mechanism still off, but the movement prediction is enabled. Now the UAVs search for targets based on their actual movement patterns, decreasing the chance of losing a target. The third and fourth scenarios follow the same pattern of scenarios one and two, one with no movement prediction, and the other predicting the targets' movement, but this time both using the handover mechanism. In all these four first scenarios, the targets maintain a constant speed during the whole simulation, not changing their speed under any circumstance. Tables 4 and 5 summarizes these simulation setup scenarios.

The two last scenarios are variations of the third and fourth scenarios featuring small controlled variations on the targets movement patterns. In scenarios 5 and 6, targets change their speed in $\pm$0.5m/s every 20 seconds, simulating the subtle speed changes that are likely to occur in real life situations of people walking. These two last scenarios

Table 4: Correct Behavior Tests Simulation Parameters

Simulation Time	900s
Simulation Area	500x500
Number of UAVs	5
Number of Targets	25
Target Initial Speed	random from 0 to 2m/s
Target Initial Heading	random
Target Speed Variation	random from $\pm$0m/s
(when active)	to $\pm$0.5m/s changing each 20s
UAV Speed	20m/s
Heuristic Stop Condition	125 evaluations
	(5 generation of 25 individuals)

Source: Prepared by the author

Table 5: Simulation Setup

Scenario	Handover	Movement Prediction
Scenario 1	disabled	disabled
Scenario 2	disabled	enabled
Scenario 3	enabled	disabled
Scenario 4	enabled	enabled

Source: Prepared by the author

are summarized in Table 6 along with scenarios 3 and 4 with whom they are comparable.

All these scenarios were primarily evaluated in terms of two principal parameters, the mean number of visits to a target and the mean time between visits. The first statistic shows how many times a target was visited during the simulation, or how many times an UAV found that target during the simulation. It is important to notice that once an UAV may have been allocated exclusively for the same target for a long time, a low visit count is not necessarily a bad result. The second statistic is important to know how frequent these visits were, or how well distributed they were. A target with a low visit count, but higher time between visits, for example, may have gotten less coverage than one with a lower interval of visits, meaning that the last had an UAV allocated to it for some time while the other one did not.

For each scenario, 100 simulations runs were performed and the presented results express an average from these runs. Table 7 shows these results. Comparing the obtained results from scenarios one and two, it can be verified that the movement prediction mechanism activated on scenario 2 works efficiently to improve target coverage, augmenting the visit count in 29% and reducing the time between visits in almost 6 seconds. Such improvements mean that targets are visited more times and more frequently. Both improvements are important for the monitoring mission. These enhancements were already expected since the movement prediction mechanism leads the UAVs to intercept targets

Table 6: Simulation Setup

Scenario	Handover	Movement Prediction	Speed
Scenario 3	enabled	disabled	constant
Scenario 4	enabled	enabled	constant
Scenario 5	enabled	disabled	variable
Scenario 6	enabled	enabled	variable

Source: Prepared by the author

based on their movement patterns, avoiding the misses that would occur if the UAVs went looking for targets on their last known position.

Table 7: Algorithm Evaluation Simulation Results

Scenario	Average Visit Count	Std. Dev Visit Count	Average TBV (s)	Std. Dev. TBV	Average Handover Count
Scenario 1	26.0	14.1	33.7	18.8	0
Scenario 2	33.6	17.0	27.8	12.3	0
Scenario 3	31.5	13.8	29.1	18.2	19
Scenario 4	35.9	16.7	25.4	12.1	10
Scenario 5	23.9	20.2	36.7	10.4	137
Scenario 6	24.4	13.2	34.2	12.3	100

Source: Prepared by the author

Notes: TBV: Time Between Visits

Analyzing scenarios 3 and 4, that have the handover mechanism activated, it is possible to observer significant gains in coverage compared to the results with scenarios 1 and 2. Comparing scenarios 1 and 3, both with no movement prediction, an improvement of 20% can be seen in visit count statistics when the handover is activated. The handover activation also causes a reduction of 4.5 seconds in the time between visits, confirming the expected improvement on coverage due to the redistribution of the targets between the UAVs. Similarly, comparing scenarios 2 and 4, both featuring movement prediction techniques, an improvement of 6% on visit count and less 2.4s on the time between visits confirm the handover positive effects on the target coverage.

Also, comparing scenarios 3 and 4 between them, the last one featuring movement prediction, the same expected positive effect caused by this mechanism can be verified. Targets on scenario 4 receive 4.4 more visits per simulation and are visited 3.7s more frequently than in scenario 3, reinforcing the idea that few targets are missed when the prediction mechanism is on.

Looking into the results from scenarios 5 and 6 it can be seen that, as expected, these simulations feature both few visits to the targets and greater times between visits when

compared to 3 and 4. This result was expected because on these simulations a lot of misses occur due to the changes in the target speeds that are unknown to the UAVs and their movement prediction and genetic algorithms, making them look for targets on the wrong locations, or calculate a bad visit order. Scenario 5, for example, performs 7.5 less visits to targets than scenario 3, while scenario 6 performs 11.5 less visits than scenario 4, showing that even subtle changes in the movement pattern of the targets may result in serious degradation on the solutions performance.

For comparison purposes, considering the worst case scenario in which a moving target changes its average speed on ± 0.5m/s between any two visits, the maximum allowed time betweeen visits so that a UAV will still find a target within it's FOV on the predicted position, was calculated to be 30s. Considering that assumption, the ideal case would be reducing the time between visits to less than 30s, meaning that no target would ever be lost if all targets respect the movement assumptions made in the problem definition section.

In order to illustrate the distribution of target to the UAVs, Figure 12 shows an example of how the UAVs (represented by the big dots circumscribed by a circle representing their FOV) spatially organize themselves and calculate the best routes to visit the targets (represented by the small dots in the figure). This a screen shot from a simulation of scenario four, and it is possible to observe that with the movement prediction turned on during that simulation, the path of the UAV is, sometimes, pretty distant from a target's current position, pointing to a future position of that target.

6.2.2 Scalability Analysis

In order to evaluate how the number of targets per UAV reflects on the solution performance, scenarios 7 to 13 were designed changing the number of targets from 20 to 100, or, in other words, 4 to 20 targets per UAV. The setup of these scenarios regarding the number of targets and which algorithms are enabled is shown in Table 8.

Table 8: Variable Target Count Simulation Setup

Scenario	Number of Targets	Handover	Movement Prediction	Target Speed
Scenario 7	20	enabled	enabled	constant
Scenario 8	25	enabled	enabled	constant
Scenario 9	30	enabled	enabled	constant
Scenario 10	35	enabled	enabled	constant
Scenario 11	40	enabled	enabled	constant
Scenario 12	60	enabled	enabled	constant
Scenario 13	100	enabled	enabled	constant

Source: Prepared by the author

Table 9 shows the results obtained from 50 runs of these scenarios, presenting an average from these runs. It can be seen from these results that, as expected, the scenarios with less targets presented increased performance on both number of visits and average time between visits. These results also showed a trend in how well the system performs when a high number of targets is confined to a small area. Observing the results from scenarios 10 and 11, it can be seen that no performance reduction was observed between

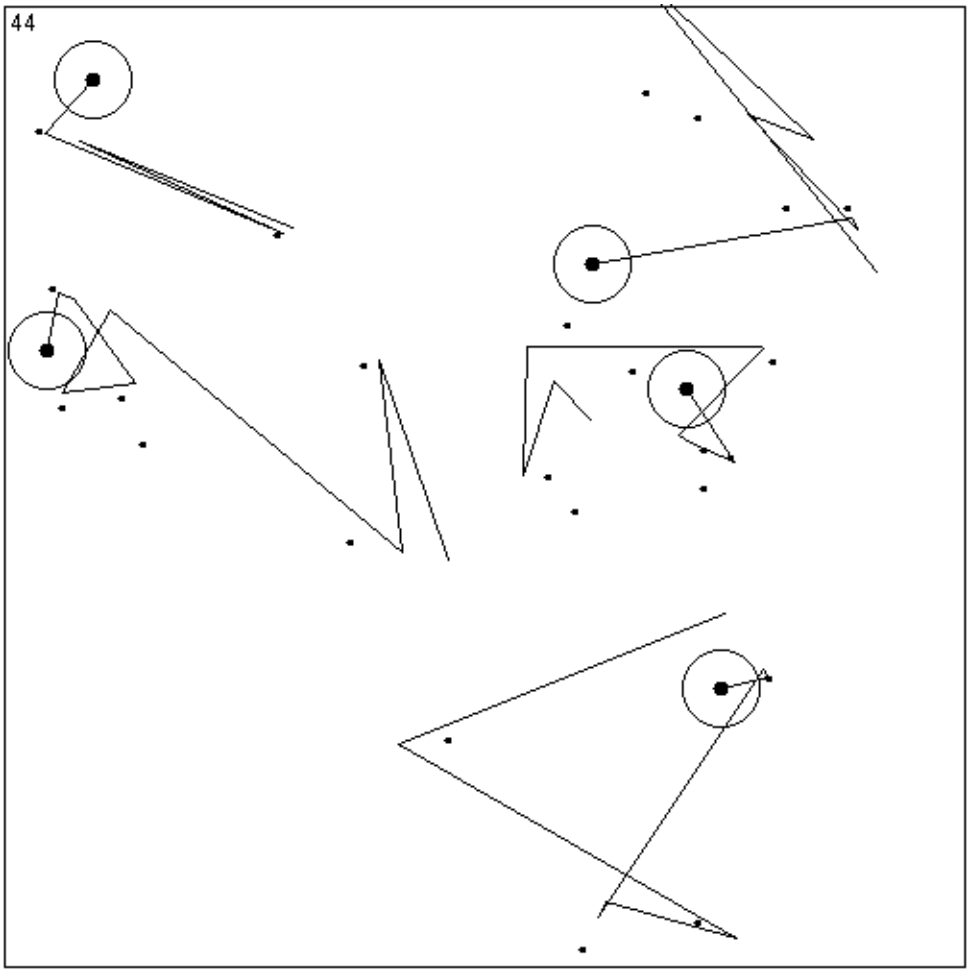


Figure 12: Target Visit Paths in a Simulation Run from Scenario 4

Source: Prepared by the author

these two cases. In fact, the performance drop from scenario 10 to scenario 11 was observed to be minimal. Further inspection comparing these simulations showed that this result was observed due to the relation between the number of targets and the simulation space. According to the observed results, confining more than 35 targets in 250,000m^2 of simulation area led the targets to be packed close together, meaning that, in average, tours that visit 7 or 8 targets per UAV are not very different, causing the average time between visits to be the same.

Moreover, in packed crowds, the simple movement of an UAV searching for a target may cause it to run into another target, visiting it by hazard. This fact ends by increasing the observation parameter of the system as a whole, reducing the average time between visits and increasing the visit count. This behavior can be well observed analyzing the results of scenarios 12 and 13, in which even a great number of targets per UAV, i.e. 20, was not able to degrade the system results below a certain level. These results, however, will not be true to a case in which targets are not limited to such a restricted area. In larger environments, where targets are not so restrained and are free to move at will, these hazards visits would not occur (or rarely occur), meaning that such high target counts might indeed degrade the system until it is not capable to observe multiple targets

anymore. However, this is a different type of scenario, compared to the one aimed to be tackled in this work.

Table 9: Variable Target Count Simulation Results

Scenario	Number of Targets	Average Visit Count	Std. Dev Visit Count	Average TBV (s)	Std. Dev. TBV
Scenario 7	20	36.05	16.70	24.97	12.70
Scenario 8	25	33.81	17.88	26.67	14.75
Scenario 9	30	30.65	14.14	29.41	14.86
Scenario 10	35	29.22	13.58	30.88	14.74
Scenario 11	40	29.24	13.73	30.88	15.51
Scenario 12	60	30.01	15.58	29.99	13.15
Scenario 13	100	30.04	24.93	29.95	15.21

Source: Prepared by the author

Notes: TBV: Time Between Visits

In addition to these results, scenarios 14, 15, 17 and 18 were designed to verify how the system deals with the arrival of new targets. Basically, as shown in Table 10, these scenarios start with 20 target and new targets, 5 or 10 of them, are inserted either 1 by 1, evenly spaced during 5 minutes of simulation (300s), or all packed together at 150s. In all these scenarios target speeds are considered constant, and both movement prediction and handover algorithm are enabled, ensuring that the best performance is always achieved. Scenarios 16 and 19, which do not spawn new targets, but start already with the final target count, were also introduced as a baseline for comparisons. All these scenarios were run 50 times each and the results hereby presented were obtained considering the average metrics from all these runs.

Table 10: Spawning Targets Simulation Setup

Scenario	Number of Initial Targets	Number of Spawning Targets	Spawning Technique	H	MP	Target Speed
Scenario 14	20	5	1 by 1	E	E	C
Scenario 15	20	5	pack	E	E	C
Scenario 16	25	-	none	E	E	C
Scenario 17	20	10	1 by 1	E	E	C
Scenario 18	20	10	pack	E	E	C
Scenario 19	30	-	none	E	E	C

Source: Prepared by the author

Notes: H: Handover; MP: Movement Prediction; E: Enabled; C: Constant

Analyzing the results presented in Table 11, it can be observed that scenarios that spawn targets during execution present, on average, better performance than those that already start with all the targets. Such behavior can be easily explained since those scenarios that start with 20 and spawn new targets, i.e. 5, have fewer targets to deal with

on the beginning, presenting a good initial performance that slowly degrades as new targets are inserted. This behavior is even more remarkable when the cases with 1 by 1 and packed insertion are compared. Pack insertion presents high visits counts during simulation, and consequently an overall reduced time between visits, it can be assessed from the results that this interval is significantly increased after new targets are spawned. The 1 by 1 insertion, on the other hand, presents a graceful degradation in performance after targets are inserted. This is an expected behavior since the targets are incrementally added, instead of abruptly inserted. These results show that the system is able to deal with new targets as their appear, trying always to minimize the effects of these situations on the overall performance.

Table 11: Spawning Targets Simulation Results

Number of Targets	25			30		
Target Spawn Technique	None	1 by 1	Pack	None	1 by 1	pack
Average Visit Count	11.2	10.4	11.4	10.2	9.6	10.6
Average Time Between Visits (Overall)(s)	26.7	26.4	24.6	29.4	27.3	25.38
Average Time Between Visits (After Spawn) (s)	-	26.3	28.9	-	29.1	29.8

Source: Prepared by the author

6.2.3 Experiments Considering Ordinary Urban Walking Scenario

Once the system was primarily designed to be used in urban environments, scenario 20 was developed to verify how the system handles a group of targets walking on a large urban avenue. The scenario is parametrized to simulate a 1500m x 50m city avenue, with 25 pedestrian targets moving along it. The targets were programmed to adopt simple ordinary human mobility behaviors, moving longitudinally along the avenue with different speeds and movement patterns. Table 12 provides details on how the simulation scenario was set-up, highlighting both UAVs' and targets' characteristics, showing which techniques and mechanisms of the algorithm are enabled.

Table 12: Scenario 20 Simulation Setup Parameters

Simulation Area	1500m x 50
Number of UAVs	5
Number of Targets	25
Target Initial Speed	random from 0.5m/s to 2.0m/s
Target Moving Pattern	Human Mobility Behavior
UAV Speed	20m/s
Handover	Enabled
Movement Prediciton	Enabled

Source: Prepared by the author

For this scenario, 50 different simulation runs were executed. In all these runs, as can be seen in Table 13, a 100% target observation coverage was verified, meaning that every

time an UAV went after a target, it was able to generate a tour to find it, and actually find it where it was expected, not losing a single scheduled observation. It can also be seen, looking at these results, that the system has achieved a high visit count, visiting each target more than 70 times in 15 minutes, with an average interval of 11.76 seconds between visits. These results corroborate the idea that this system is highly able to efficiently perform in urban law enforcement tasks, i.e. demonstration or street party monitoring, completely fulfilling its primary goal. Moreover, notice that the results reported in this section are even better than those previous reported in the sections above. These better results are understandable due to the spacial distribution of the targets in this last scenario, that force them to stay closer, compared to those areas considered before.

Table 13: Scenario 20 Simulation Results

Average Visit Count	76.50
Std. Dev. Visit Count	17.90
Average Time Between Visits (s)	11.76
Std. Dev. Time Between Visits (s)	9.42
Average Handover Count per Target	2.04
Target Observation Coverage	100%

Source: Prepared by the author

6.3 Design Space Exploration

In order to test the developed heuristic methods, a set of tests was elaborated to evaluate the performance of each optimization method to solve the problem focusing on finding the best solution. Once in real life situations the intended system will mostly be used to intercept just a few targets, and not hundreds of them, thus each method was tested against sets of 8, 12, 16 and 20 randomly placed targets, with speeds ranging from 0.5 m/s to 2 m/s in 0.5m/s steps (a reasonable range for human walking speed). The interceptor's, or traveler, speed was set to 20m/s throughout all the simulations (a reasonable speed for a small surveillance COTS UAV). To avoid a significant influence of the different parametrization on the different heuristic algorithms, a limit of 10,000 maximum solutions tested was set for comparison purposes. This choice ensures that, for every different algorithm, the same number of candidate solutions will be tested. It is important to notice that the fitness function, or function being minimized on this work is not the total lenght of the tour, but the average displacement of the targets during the tour, as defined in (15) and (16).

Table 14 summarizes the experiment parametrization, while Tables 15 and 16 present the obtained results.

As seen in Table 15 both GA and ACO present equally promising results on minimizing the displacement of the targets on the tour regardless of the number or speed of the targets. It can be verified, however, that once the number of targets starts to increase, the genetic algorithm starts to present a slightly better performance, outperforming both ACO and SA algorithms. The ant colony optimization (ACO) presents good results for small numbers of targets, sometimes finding the same optimal results as the GA, but as the number of targets increases, the number of generations necessary to find good solutions extrapolates the limit of 10,000 comparisons, influencing the results. On the other hand, the simulated Annealing algorithm sometimes manages to find good solutions, but,

Table 14: Tour Length Performance Testing Parameters

Number of Targets	8,12,16,20
Interceptor Speed	20 m/s
Target Speeds	0.5m/s 1m/s, 1.5 m/s, 2 m/s
Target Distribution Area	Square 600m x 600
Max Number of Solutions Tested	10,000
Number of Testing Iterations	250
Fitness Function	Maximum Displacement of the Targets on the Tour

Source: Prepared by the author

Table 15: Performance of the Heuristic Algorithms on Minimizing the Displacement of The Targets visited in a Tour (m)

	8 Targets		12 Targets		16 Targets		20 Targets	
	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Genetic Algorithm								
0.5 m/s	8.42	8.60	5.65	6.33	6.65	7.12	4.77	6.20
1.0 m/s	13.13	13.73	13.15	15.30	10.60	13.09	9.99	13.72
1.5 m/s	21.6	23.34	20.5	23.37	22.81	26.22	14.57	18.61
2.0 m/s	27.43	29.72	22.79	26.33	25.91	31.53	18.18	26.55
Simulated Annealing								
0.5 m/s	9.34	12.33	8.69	13.03	9.31	15.17	11.22	14.98
1.0 m/s	14.52	27.13	18.99	29.55	18.32	32.87	21.28	35.97
1.5 m/s	26.04	42.35	35.47	55.16	34.96	54.05	34.6	60.94
2.0 m/s	32.29	57.43	34.53	69.04	41.60	111.05	46.32	101.7
Ant Colony Optimization								
0.5 m/s	8.42	8.46	5.65	6.02	6.72	7.02	5.05	6.18
1.0 m/s	13.13	13.57	13.24	15.43	10.60	12.26	9.79	13.59
1.5 m/s	21.60	22.89	20.53	21.70	22.81	25.5	14.58	20.89
2.0 m/s	27.43	28.05	22.81	23.40	26.64	33.81	19.26	30.83

Source: Prepared by the author

Notes: Table presents the minimum and average displacement of the targets during the simulations in meters

on average, presents a much worse performance than GA and ACO.

It is noticeable from the results that, as the movement speeds of the targets increase, the average displacement of the targets also tends to increase. This result was already expected since faster targets tend to move more in the same time interval.

Comparing the temporal performance of the algorithm using the results presented in Table 16, the advantages of the Genetic Algorithm become even more evident. Besides faster, the SA algorithm was observed to be the worst of them on average, while the ACO, in its turn, presents good results for lower numbers of targets, but as this number increases, it becomes the most time-consuming of the three. Another important characteristic to be noticed is that speed has a significant effect on the temporal behavior of the algorithms. It

Table 16: Temporal Performance of the Heuristic Algorithms on Solving the TD-TSP Problem

	8 Targets	12 Targets	16 Targets	20 Targets
Genetic Algorithm				
0.5 m/s	57.89ms	80.04ms	98.08ms	119.14ms
1.0 m/s	62.55ms	85.64ms	104.66ms	128.86ms
1.5 m/s	63.18ms	88.39ms	109.87ms	135.81ms
2.0 m/s	67.39ms	93.35ms	118.71ms	142.55ms
Simulated				
0.5 m/s	30.00ms	37.94ms	49.15ms	59.04ms
1.0 m/s	28.93ms	41.63ms	54.73ms	66.33ms
1.5 m/s	30.44ms	45.12ms	56.15ms	68.93ms
2.0 m/s	32.57ms	45.96ms	61.31ms	75.06ms
Ant Colony	Optimization			
0.5 m/s	57.33ms	103.22ms	169.08ms	247.59ms
1.0 m/s	59.32ms	108.42ms	169.97ms	252.08ms
1.5 m/s	60.60ms	108.26ms	174.92ms	255.18ms
2.0 m/s	61.80ms	114.32ms	181.06ms	265.26ms

Source: Prepared by the author

can be observed that all execution times increased with the targets' speed.

From these results, it can be accessed that the best algorithm to be used in the TD-TSP problem that tries to minimize the displacement of the targets is the Genetic Algorithm, which has shown both good behavioral and temporal performance solving the addressed problem.

7 CONCLUSION AND FUTURE WORK

7.1 Summary

This work has presented a proposal that enables UAVs to periodically monitor a group of moving targets that simulate the movement of walking individuals in crowded environments. The goal on developing this work was investigating how well a group of UAVs can continuously monitor a large group of individuals (targets) in a crowd, alternatingly visiting each of them at a time while trying to not lose sight of any of these targets. A system equipped with a group of UAVs running this proposal can be used for law-enforcement applications, assisting authorities to monitor crowds in order to identify and follow suspicious individuals that can have attitudes that could be classified as vandalism or linked to terrorist attack attempts.

In order to correctly address the problem, an auction algorithm was put in place to distribute interest targets among the multiple UAVs, which, in turn, made use of a genetic algorithm to calculate the order in which they would visit each of the targets on their observation queue. Moreover, a target handover algorithm was also implemented to redistribute targets among the UAVs in case the system judged that a target was about to be lost by its current observer UAV.

Simulations performed to test the developed approach have shown that it was able to perform such monitoring task, visiting the targets during the simulations, while still minimizing the time between visits made to these targets. According to these simulations, the genetic algorithm has shown a great potential to solve this kind of moving target monitoring and TDTSP problems, and the implemented handover mechanism was able to handle target redistribution assertively, maximizing the system performance. The results have shown that a group of 5 UAVs can monitor up to 25 moving targets on a 1500m avenue visiting each one more than 76 times in 15 minutes, which means a visit every 12 seconds to most targets, fulfilling completely its intended goal of observing multiple targets to verify their behavior on a regular basis.

Overall, considering the outlined context, the implemented solution has presented a solid and satisfactory behavior when it comes to addressing the challenges proposed. Moreover, the results of this work have shown that COTS UAVs can be coordinated to become a viable tool for crowd monitoring applications, supporting law enforcement operations in urban or restricted environments.

7.2 Further Work

Proposed future works on this project comprehend the analysis and study of the genetic algorithm and its parameters, such as the number of generations, genome, and popu-

lation size, fitness function and fitness analysis to try to enhance the system performance. These enhancements could be focused on augmenting the number of visits to the targets and reducing the time between these visits. Another important addition to the work would be the adaptation of the auction and handover mechanism. This adaptation would be mainly about tweaking the bid rules, to reduce the number of handovers performed during the system execution and the best allocation of the targets.

Additionally, extended simulation scenarios can be analyzed, such as those in which targets follow different patterns of restrict movement model, studying a larger variety of movement patterns of crowds taking part in demonstrations or riots. For these scenarios a stochastic or predictive target interception method that considered the environment structure could also be proposed. This amelioration could consider the proximity to street intersections, for example, as a factor to be looked at when scheduling the visits to a target, observing it when it reaches intersections and avoiding losing it if it turns left or right on a street. Likewise, new movement interception methods could be explored and their impacts on the performance evaluated. Interception methods could also be considered as a system parameter, allowing the human controller to decide how the UAVs will try to intercept the targets, making the system more dynamic since this choice could be made taking specific scenario constraints, such as the environment, into account.

Moreover, it could be interesting to analyze how communication losses or failures influence the system. Supposing that an UAV takes a turn on a different street and loses communication with the rest of the swarm, it is yet unknown how the system as a whole would react. To avoid degradation in such situations some mechanisms that redistribute the targets of the disconnected UAV among the connected ones could be studied.

Furthermore, as both the proposed approach and the computer vision module, which is considered transparent to this system, have temporal constraints and need to interact, a performance analysis needs to be made running both modules on a real computation platform to investigate how their interaction affect the system. Both modules running on the same processor can, for example, slow down the system as a whole, degrading its performance and causing it to lose a target. In another possible situation, the pattern recognition module, which recognizes the targets, may take too long to process a frame or be too slow to interact with the coordination module, not registering a visit to a target. Due to these possibilities, it would be imperative to conduct a real-time performance evaluation on the system as a whole to test the performance of the integrated modules working together.

Also, as the proposed system can be quite mission-critical depending on the application, its safety and security constraints also deserve to be investigated. In some situations, for example, terrorist or rioters can try to high-jack or attack the UAVs to cover the tracks of their criminal activities, sending the UAVs away from a given target or even bringing the system or the UAVs down. To prevent safety and security breaches, future strategies could be employed on this system to make it less vulnerable to failures. As an example, one simple option could be the addition of a cryptography module, increasing the security of the communication between drones. In addition, to provide an additional safety solution, a centralized computation module could be added and used to coordinate the whole system under normal conditions, this way the distributed system could be activated locally on the UAVs when necessary, making the system more reliable, robust, and energy efficient since the processing would take place mainly on the central module and not on the UAVs themselves.

This work could also be further expanded by means of a target priority mechanism,

marking some targets as more important than others, and thus helping the system decide on which targets to handover and on how to act in case of an unavoidable degradation. This mechanism would help, for example, in situations in which a group of targets could not be fully visitable at some point during the simulation, meaning that some target would have to be either handed over or completely removed from the list of targets, being no longer observed. These situations are very likely to occur in urban scenarios, where some targets can turn left or right in a corner, where other targets continue moving ahead, making it necessary for the system to decide what to do and wether to keep observing these two different groups of targets, a situation where assigning a priority to either of the groups, or some targets within them, would help.

REFERENCES

3DROBOTICS. **Iris Plus Quadcopter**. Available at: <http://3dr.com/support/articles/207358106/iris>. Visited on: February, 2nd, 2018.

ABDOUN, O.; ABOUCHABAKA, J.; TAJANI, C. Analyzing the performance of mutation operators to solve the travelling salesman problem. **International Journal of Emerging Sciences (IJES)**, [S.l.], v.2, n.1, p.61-77, 2012.

ABELEDO, H. et al. The time dependent traveling salesman problem: polyhedra and algorithm. **Mathematical Programming Computation**, Heidelberg, v.5, n.1, p.27-55, Mar. 2013.

ADAMEY, E.; OZGUNER, U. A decentralized approach for multi-UAV multitarget tracking and surveillance. In: GROUND/AIR MULTISENSOR INTEROPERABILITY, INTEGRATION, AND NETWORKING FOR PERSISTENT ISR III, 2012, Baltimore. **Proceedings...** Bellingham: International Society for Optics and Photonics (SPIE), 2012. v.8389, p.838915-838915.

AKNINE, S.; PINSON, S.; SHAKUN, M. F. An extended multiagent negotiation protocol. **Autonomous Agents and MultiAgent Systems**, Boston, v.8, n.1, p.5-45, 2004.

APPLEGATE, D. **The traveling salesman problem**:a computational study. Princeton: Princeton University Press, 2006. (Princeton Series in Applied Mathematics).

ARRAM, A.; AYOB, M.; ZAKREE, M. Comparative study of metaheuristic approaches for solving traveling salesman problems. **Asian Journal of Applied Sciences**, [S.l.], v.7, n.7, p.662-670, 2014.

BERTSEKAS, D. P. Auction algorithms for network flow problems: a tutorial introduction. **Computational Optimization and Applications**, Boston, v.1, n.1, p.7-66, 1992.

BURKERT, F.; FRAUNDORFER, F. Uav-based monitoring of pedestrian groups. **International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences ISPRS**, Gottingen, n.2, p.67-72, Aug. 2013.

CAI, Z. et al. Multirobot cooperative pursuit based on task bundle auctions. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTICS AND APPLICATIONS, 2008, Wuhan. **Proceedings...** Berlin: Springer, 2008. p.235-244.

CHAPRA, S. C.; CANALE, R. P. **Numerical methods for engineers**. New York: McGraw-Hill, 1998. v.2.

CHEN, H.; CHANG, K.; AGATE, C. S. UAV path planning with tangent-plus-Lyapunov vector field guidance and obstacle avoidance. **IEEE Transactions on Aerospace and Electronic Systems**, New York, v.49, n.2, p.840-856, Apr. 2013.

CHENG, C.; MAO, C. A modified ant colony system for solving the travelling salesman problem with time windows. **Mathematical and Computer Modelling**, Oxford, v.46, n.9, p.1225 - 1235, 2007.

CHOUBEY, N. S. Moving target travelling salesman problem using genetic algorithm. **International Journal of Computer Applications**, [Bangalore], v.70, n.2, p.30-34, 2013.

COSTA, M. Interpersonal distances in group walking. **Journal of Nonverbal Behavior**, Dordrecht, v.34, n.1, p.15-26, 2010.

DAVIS, L. Applying adaptive algorithms to epistatic domains. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 9., 1985, Los Angeles. **Proceedings...** San Francisco: Morgan Kaufmann Publishers Inc., 1985. p.162-164.

DEKKERS, A.; AARTS, E. Global optimization and simulated annealing. **Mathematical Programming**, Berlin, v.50, n.1, p.367-393, Mar. 1991.

DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization. **IEEE Computational Intelligence Magazine**, New York, v.1, n.4, p.28-39, Nov. 2006.

DORIGO, M.; CARO, G. D. Ant colony optimization: a new metaheuristic. In: CONGRESS ON EVOLUTIONARY COMPUTATION - CEC99, 1999, Washington. **Proceedings...** [New York]: IEEE, 1999. v.2, p.1477 Vol. 2.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B**, Piscataway, v.26, n.1, p.29-41, Feb. 1996.

EASLEY, D.; KLEINBERG, J. **Networks, crowds, and markets**: reasoning about a highly connected world. New York: Cambridge University Press, 2010.

ELMOGY, A. M.; KARRAY, F. O. Cooperative multi target tracking using multi sensor network. **International Journal on Smart Sensing and Intelligent Systems**, [S.l.], v.1, n.3, p.716-734, 2008.

ENGLOT, B.; SAHAI, T.; COHEN, I. Efficient tracking and pursuit of moving targets by heuristic solution of the traveling salesman problem. In: IEEE CONFERENCE ON DECISION AND CONTROL, 52., 2013, Firenze. **Proceedings...** New York: IEEE, 2013. p.3433-3438.

FANG, Z.; LO, S.; LU, J. On the relationship between crowd density and movement velocity. **Fire Safety Journal**, Lausanne, v.38, n.3, p.271-283, 2003.

FROHNWIESER, A. et al. Human walking behavior: the effect of pedestrian flow and personal space invasions on walking speed and direction. **Human Ethology Bulletin**, Seattle, v.28, n.3, p.20-28, 2013.

GERKEY, B. P.; MATARIC, M. J. Sold!: auction methods for multirobot coordination. **IEEE Transactions on Robotics and Automation**, New York, v.18, n.5, p.758-768, Oct. 2002.

GHANDESHTANI, K. S. et al. Dynamic local search algorithm for solving traveling salesman problem. In: INTERNATIONAL CONFERENCE ON ADVANCED ENGINEERING COMPUTING AND APPLICATIONS IN SCIENCES, ADVCOMP 2010, 4., 2010, Florence. **Proceedings...** [Wilmington]: International Academy: Research: and Industry Association (IARIA), 2010.

GOLDBERG, D. E.; LINGLE JR., R. Alleles, loci, and the traveling salesman problem. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 1., 1985, Pittsburgh. **Proceedings...** Hillsdale: L. Erlbaum, 1985. p.154-159.

GREFENSTETTE, J. et al. Genetic algorithms for the traveling salesman problem. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 1., 1985, Pittsburgh. **Proceedings...** Hillsdale: L. Erlbaum, 1985. p.160-168.

GREFENSTETTE, J. J. Incorporating problem specific knowledge into genetic algorithms. In: DAVIS, L. (Ed.). **Genetic algorithms and simulated annealing**. London: Pitman Publishing, 1987. p.42-60.

GUTIN, G.; PUNNEN, A. P. (Ed.). **The traveling salesman problem and its variations**. Dordrecht: Springer US, 2002. (Combinatorial Optimization).

HELBING, D. et al. Selforganizing pedestrian movement. **Environment and planning B: planning and design**, London, v.28, n.3, p.361-383, 2001.

HELVIG, C.; ROBINS, G.; ZELIKOVSKY, A. The movingtarget traveling salesman problem. **Journal of Algorithms**, San Diego, v.49, n.1, p.153-174, 2003.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. 2.ed. Cambridge: MIT Press, 1992.

ILAVARASI, K.; JOSEPH, K. S. Variants of travelling salesman problem: a survey. In: INTERNATIONAL CONFERENCE ON INFORMATION COMMUNICATION AND EMBEDDED SYSTEMS (ICICES2014), 2014, Chennai. **Proceedings...** Piscataway: IEEE, 2014. p.1-7.

KELLER, J. **Wrong again**: uav market heading nowhere but up over the next decade. Available at: <http://www.militaryaerospace.com/articles/2015/08/uav-market-blog.html>. Visited on: February, 2nd, 2018.

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

KIM, J.; CRASSIDIS, J. L. UAV path planning for maximum visibility of ground targets in an urban area. In: INTERNATIONAL CONFERENCE ON INFORMATION FUSION, 13., 2010, Edinburgh. **Proceedings...** Piscataway: IEEE, 2010. p.1-7.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, New York, v.220, n.4598, p.671-680, 1983.

KOLLING, A.; CARPIN, S. Cooperative observation of multiple moving targets: an algorithm and its formalization. **The International Journal of Robotics Research**, Cambridge, v.26, n.9, p.935-953, 2007.

KRETZ, T. Pedestrian traffic: on the quickest path. **Journal of Statistical Mechanics: theory and experiment**, Bristol, v.2009, n.03, p.3012-3038, 2009.

LAFREE, G.; DUGAN, L. Evolution of global terrorism. In: BACKER, D.; BHAVNANI, R.; HUTH, P. (Ed.). **Peace and conflict 2017**. New York: Routledge, 2017. (Peace and Conflict).

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling salesman problem. **Operations Research**, Baltimore, v.21, n.2, p.498-516, 1973.

LINCHANT, J. et al. Are unmanned aircraft systems (UASs) the future of wildlife monitoring? a review of accomplishments and challenges. **Mammal Review**, [S.l.], v.45, n.4, p.239–252, 2015.

MATAI R, S. S.; MITTAL, M. L. Traveling salesman problem: an overview of applications, formulations, and solution approaches. In: DAVENDRA, D. (Ed.). **Traveling salesman problem, theory and applications**. [S.l.]: InTech, 2010.

MENEZES, F. M.; MONTEIRO, P. K. **An introduction to auction theory**. Oxford: Oxford University Press, 2005.

MORAES, R. S. de; FREITAS, E. P. de. Distributed control for groups of unmanned aerial vehicles performing surveillance missions and providing relay communication network services. **Journal of Intelligent & Robotic Systems**, Dordrecht, Nov. 2017.

NATALIZIO, E. et al. Two families of algorithms to film sport events with flying robots. In: INTERNATIONAL CONFERENCE ON MOBILE AD-HOC AND SENSOR SYSTEMS, 10., 2013, Hangzhou. **Proceedings...** Piscataway: IEEE, 2013. p.319-323.

NELSON, H.; MACLENNAN, H. Emergency movement. In: Society of Fire Protection Engineers (Ed.). **Society of fire protection engineering handbook**. Massachusetts: Society of Fire Protection Engineers, 1995.

OH, H. et al. Coordinated standoff tracking of moving target groups using multiple UAVs. **IEEE Transactions on Aerospace and Electronic Systems**, New York, v.51, n.2, p.1501-1514, Apr. 2015.

PACK, D. J. et al. Cooperative control of UAVs for localization of intermittently emitting mobile targets. **IEEE Transactions on Systems, Man, and Cybernetics, Part B**, Piscataway, v.39, n.4, p.959-970, Aug. 2009.

PARKER, L. E.; EMMONS, B. A. Cooperative multirobot observation of multiple moving targets. In: INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1997, Albuquerque. **Proceedings...** Piscataway: IEEE, 1997. v.3, p.2082-2089.

POTVIN, J. Genetic algorithms for the traveling salesman problem. **Annals of Operations Research**, Basel, v.63, n.3, p.337-370, Jun. 1996.

RAMAN, V.; GILL, N. S. Review of different heuristic algorithms for solving travelling salesman problem. **International Journal of Advanced Research in Computer Science**, [India], v.8, n.5, p.423-425, 2017.

RAZALI, N. M.; GERAGHTY, J. Genetic algorithm performance with different selection strategies in solving TSP. In: WORLD CONGRESS ON ENGINEERING, 2011, London. **Proceedings...** [S.l]: Newswood, 2011. v.2, p.1134-1139.

SHUKLA, A.; PANDEY, H.; MEHROTRA, D. Comparative review of selection techniques in genetic algorithm. In: INTERNATIONAL CONFERENCE ON FUTURISTIC TRENDS IN COMPUTATIONAL ANALYSIS AND KNOWLEDGE MANAGEMENT, ABLAZE, 1., 2015, Greater Noida. **Proceedings...** [Piscataway]: IEEE, 2015.

SMITH, R. G. The contract net protocol: high-level communication and control in a distributed problem solver. **IEEE Transactions on computers**, [New York], n.12, p.1104-1113, 1980.

STÜTZLE, T.; HOOS, H. H. MAX-MIN ant system. **Future Generation Computer Systems**, [Amsterdam], v.16, n.8, p.889 - 914, 2000.

SU, Z. C.; HLAING, S. H.; KHINE, M. A. An ant colony optimization algorithm for solving traveling salesman problem. In: INTERNATIONAL CONFERENCE ON INFORMATION COMMUNICATION AND MANAGEMENT, 1., 2011, Singapore. **Proceedings...** [S.l.: s.n.], 2011. v.16, p.54-59.

SUJIT, P. B.; BEARD, R. Distributed sequential auctions for multiple UAV task allocation. In: AMERICAN CONTROL CONFERENCE, 2007, New York. **Proceedings...** [New York]: IEEE, 2007. p.3955-3960.

SYSWERDA, G. Schedule optimization using genetic algorithms. In: DAVIS, L. (Ed.). **Handbook of genetic algorithms**. New York, NY: Van Nostrand Reinhold, 1991.

TUBA, M.; JOVANOVIC, R.; JOVANOVIC, R. Improved ACO algorithm with pheromone correction strategy for the traveling salesman problem. **International Journal of Computers, Communications and Control (IJCCC)**, Oradea, v.8, p.477-485, Apr. 2013.

ÜÇOLUK, G. Genetic algorithm solution of the TSP avoiding special crossover and mutation. **Intelligent Automation & Soft Computing**, Albuquerque, v.8, n.3, p.265-272, 2002.

VICKREY, W. Counterspeculation, auctions, and competitive sealed tenders. **The Journal of Finance**, [New York], v.16, n.1, p.8--37, 1961.

WALSH, W. E.; WELLMAN, M. P. A market protocol for decentralized task allocation. In: INTERNATIONAL CONFERENCE ON MULTI AGENT SYSTEMS, 1998, [S.l.]. **Proceedings...** [S.l.]: IEEE, 1998. p.325-332.

WHITLEY, L. D.; STARKWEATHER, T.; FUQUAY, D. Scheduling problems and traveling salesmen: the genetic edge recombination operator. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS ICGA, 3., 1989, Fairfax. **Proceedings...** San Francisco: Morgan Kaufmann, 1989. p.133-140.

XU, M.; LI, S.; GUO, J. Optimization of multiple traveling salesman problem based on simulated annealing genetic algorithm. In: GLOBAL CONGRESS ON MANUFACTURING AND MANAGEMENT (GCMM 2016), 13., 2017, Zhengzhou. **Proceedings...** [S.l.]: MATEC Web of Conferences, 2017. v.100, p.2025.

YADAV, P. K.; MENG, W. Mobile targets localization in a field area using moving Gaussian peaks and probability map. In: IEEE INTERNATIONAL CONFERENCE ON CONTROL AUTOMATION (ICCA), 11., 2014, Taichung. **Proceedings...** Piscataway: IEEE, 2014. p.416-421.

ZHAN, S. et al. List-based simulated annealing algorithm for traveling salesman problem. **Computational Intelligence and Neuroscience**, [New York], v.2016, p.1712630, Mar. 2016.