MULTITARGET TRACKING SYSTEM WITH CONNECTIVITY CONSTRAINTS

Rafael Lima de Carvalho

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Felipe Maia Galvão França
Félix Mora-Camino

Rio de Janeiro
Março de 2016

MULTITARGET TRACKING SYSTEM WITH CONNECTIVITY CONSTRAINTS

Rafael Lima de Carvalho

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Félix Mora-Camino, Ph.D.

Prof. Ricardo Cordeiro de Farias, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Max Suell Dutra, Dr.-Ing.

Prof. Cláudia Marcela Justel, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2016

Carvalho, Rafael Lima de

Multitarget Tracking System with connectivity constraints/Rafael Lima de Carvalho. – Rio de Janeiro: UFRJ/COPPE, 2016.

XIV, 124 p.: il.; 29, 7cm.

Orientadores: Felipe Maia Galvão França

Félix Mora-Camino

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 104 – 114.

1. Swarm Intelligence. 2. Optimization. 3. Relay Robotic Networks. I. França, Felipe Maia Galvão *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicado à Maria Ione, Miguel e Débora.

Agradecimentos

Agradeço inicialmente a Deus pela resiliência e força dispensadas a mim no decorrer desta longa e dificultosa jornada. Aos meus mestres e grandes amigos Felipe e Felix pelos conhecimentos e valores ensinados durante este caminho, os quais vão muito além dos conhecimentos acadêmicos. Me ensinaram a ser uma pessoa melhor e um melhor professor.

À minha família pelo apoio incondicional.

Aos amigos e colegas de Palmas, Warley Gramacho, Glendara Martins, Ary Henrique, Marcelo Leineker, Tiago Almeida, cujo apoio foi mais que fundamental para a continuação do doutorado.

Gostaria também de agradecer aos colegas de trabalho, pertencentes ao corpo docente do curso de Ciência da Computação. Agradecer aos professores Gentil, Patrick e Warley da UFT que juntamente com o apoio dos professores Xexéo e Maculan da UFRJ, tornaram o projeto Dinter-TO uma realidade.

Aos maravilhosos amigos que ganhei durante a jornada do doutorado no PESC: Saulo, Anna Laura, Raphael, Leandro Marzulo, Popov, Thiago, Flávio, Fábio, João Amarante, Douglas, Daniel Alves, Danilo, pessoas que adicionaram boas discussões e diversões no processo de doutoramento.

Por fim, aos membros da secretaria e do suporte do PESC, os quais foram sempre gentis e prestativos em todas as ocasiões.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

SISTEMA DE RASTREAMENTO DE MÚLTIPLOS ALVOS SOB RESTRIÇÕES DE CONECTIVIDADE

Rafael Lima de Carvalho

Março/2016

Orientadores: Felipe Maia Galvão França
Félix Mora-Camino

Programa: Engenharia de Sistemas e Computação

A primeira parte deste trabalho lida com o problema de posicionar um grupo de agentes retransmissores (*relays*) de forma a dar conectividade a um segundo grupo de agentes ativos (*pursuers*). A primeira abordagem apresentada consiste em modelar o cenário como um problema de programação quadrática (PPQ) com restrições lineares, usando uma estrutura de conectividade fixa. Para resolver o modelo proposto, foi implementada uma rede neural recorrente a qual converge rapidamente para a solução ótima do problema, mesmo em instâncias razoavelmente grandes. Como forma de avaliação, realizou-se um comparativo entre o *solver* de PPQ da plataforma Matlab e a rede proposta, também implementada na mesma plataforma. Na segunda abordagem foi proposto o uso de uma estimativa da conectividade algébrica do grafo de proximidade gerado pela rede, para direcionar o grupo de *relays* e *pursuers*, usando-se apenas as informações da vizinhança de cada agente. Nesta abordagem a estrutura do grafo é dinâmica, além disso, como proposta de paralelização, a esta solução distribuída foi acoplada um algoritmo de escalonamento por reversão de arestas (SER). Além do mais, as metaheurísticas *Simulated Annealing*, *Genetic Algorithms*, *Particle Swarm Optimization* e *Backtracking Search Algorithm* foram implementadas como alternativas de soluções ao problema. As soluções são avaliadas em um cenário de perseguição de alvos, os quais podem possuir comportamentos reativos (tais como fugir dos perseguidores). A segunda parte deste trabalho investiga o problema de rastreamento de múltiplos objetos em tempo real. Como solução, foi proposto um algoritmo que se baseia em memória de curto e longo prazo usando redes neurais sem peso.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

MULTITARGET TRACKING SYSTEM WITH CONNECTIVITY CONSTRAINTS

Rafael Lima de Carvalho

March/2016

Advisors: Felipe Maia Galvão França
Félix Mora-Camino

Department: Systems Engineering and Computer Science

The first part of this work deals with the problem of positioning a swarm of relay agents with the objective of providing connectivity to a second group of active agents (pursuers). The first approach consists of modelling the considered scenario as a quadratic programming problem (QP) with linear restrictions, using a fixed graph structure. In order to solve such model, a recurrent neural network is proposed with fast convergence rate to the optimal solution, even with reasonably big size instances. In addition, a comparison with the Matlab QP solver has been conducted in some experimental simulations. In the second approach, it is proposed an estimation of the algebraic connectivity of the underlying graph generated by the network. Over this estimation, it is proposed a metric to direct the group of relays and pursuers, using only local neighbourhood information of each agent. On this approach, the graph structure is dynamic and it is also proposed the use of the schedule by edge reversal (SER) as a solution to ordering the parallelization of the robot positioning computation. Moreover, the meta-heuristics *Simulated Annealing*, *Genetic Algorithms*, *Particle Swarm Optimization* and *Backtracking Search Algorithm* have been applied as alternative solution providers. The proposed solutions have been applied in a target pursuit scenario, for which the targets are deployed in different spots and may have some reactive behaviours (such as escape from the pursuers). The second part of this work investigates the visual tracking of shape shift objects in real time. As a solution, it is proposed a short- and long-time memories tracker which uses a weightless neural network for training and retraining the objects patterns.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Some tasks can be too complex to be executed by a single robot, therefore the idea of using a team of mobile robots instead of a single one has emerged into the field of swarm robotics. Using a team of less expensive robots can give more efficient, more flexible, and more fault-tolerant models than having a single highly specialized and powerful robot for each task. But at the same time, it increases the amount of difficulties on modelling and implementing distributed solutions using a group of robots. Furthermore, the distributed information and the degree of parallelism for doing complex tasks is an attractive advantage. However, the understanding and use of collective intelligence of a set of robots are still a challenge task for the computational intelligence field.

Swarm intelligence is a modern computational intelligence discipline that is concerned with the design of multi-agent systems with applications, e.g. in optimization and robotics. Instead of a sophisticated controller that governs the global behaviour of the system, the main principle of swarm intelligence is based on many unsophisticated entities that cooperate with each other in order to exhibit a desired behaviour. The design of these systems is inspired by the collective behaviour of social insects such as ants, termites, bees, and wasps. Some techniques are based on other animal societies such as flocks of birds or schools of fish [12]. More recently, biological-inspired swarm research has arisen up. In special, swarm techniques such as Particle Swarm Optimization (PSO)[13], Ant Colony Optimization [14] and Artificial Bee Colony [15] have been applied successfully in the field of optimization and robotics.

The advancement of science and engineering, as well as the price reduction of electronic components has contributed to the popularization of flying civil prototypes, for example the quadrotors. Another step towards the accessibility of such drones was the high level control interface used by some companies. As an example, about 2011, a company called Parrot Inc. released a quadrotor capable of flying using the gyroscope present in popular smartphones. The Ardrone 1.0,

as they called their prototype, was capable of creating an ad-hoc wireless network accepting connection from the smarphone, and with self stabilization for taking off and landing, the flying instructions come from the positioning of the smartphone. In the front of the Ardrone has a camera which images are streamed directly to the smartphone screen. A photo of this prototype can be seen in Figure 1.1.

Figure 1.1: A picture of the Parrot's Ardrone 1.0.

More recently, a group of researchers of MIT's GRASP Laboratory [16] developed micro quadrotor prototypes with only 75g of weight. They ran an experiment with 20 micro robots showing their agility in executing manoeuvres. As the IEEE 802.11 Ethernet device embedded in such devices, it could be realized they could be used as a network routers, and when navigating in group, a dynamic network can arise from them. Furthermore, as any other communication device, the wifi has distance limitations. Therefore, in order to use the dynamic network possibility, a control scheme using the information based on the quality of the signal should be realized as well. With a control scheme like this, the application possibilities are wide.

In 2007, unfortunately an earthquake shook Peru killing at least 437 people and leaving another 1350 injured [17]. More recently, in 2015, one of the worst mining accident happened in Mariana city, Minas Gerais state, in Brazil [18]. In such scenarios, a team of autonomous flying robots could be deployed, keeping the network with some base station and survey the area seeking for survivors, for example.

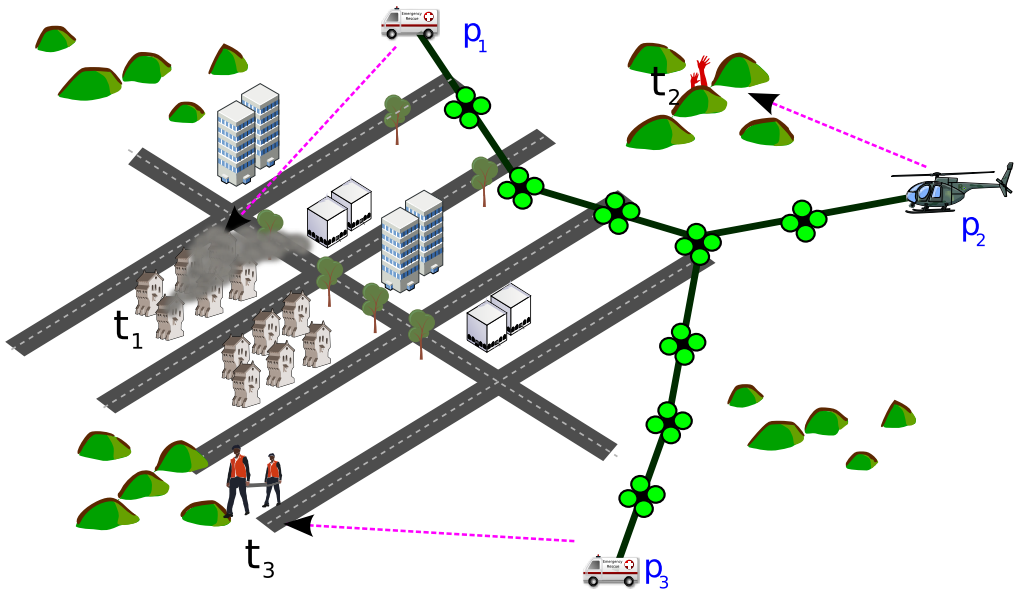
Recently Shekhar *et.al.* [19] published a review on the applications and necessity of spatial computing. Technologies such as GPS (Global Positioning System) has improved the quality of services all around the world. Since driving until the birth of Uber-like services. According to the authors, the majority of human beings spends

their time in indoor environments. Localization systems such GPS do not work well in indoor locations. This gives rise to another application: the maintenance and configuration of a temporary positioning system. Since using at least three spots, a location of an object can be triangularized, then a team of moving robots could deploy a system like that.

This thesis deals with the problem of tracking mobile targets using multiple Unnamed Aerial Vehicles (UAVs), while maintaining the connectivity of the network among them. This problem is correlated with the wireless sensor network's coverage problem, in a way that to track targets, mobile robots should stay connected, even for certain periods of time. So the network connectivity should be preserved which allow the entities exchange the acquired knowledge during the tracking task. Therefore, the main objective of this research project is divided into two subjects: a) investigate and propose global and locally distributed computational solutions for the connectivity maintenance problem; and b) produce computational efficient algorithms for video tracking of generic objects using vision sensor.

Figure 1.2 illustrate a scenario addressed by the present work. In a rescue situation, where the targets are trying to cooperate with the tracking, they try to move towards the pursuers, as depicted by Figure 1.2. Another surrogate scenario is when the targets are trying to escape from the pursuers. In such scenario, the instance problem can be difficult to solve, since the limited number of entities imposes limitations on the reachable areas the pursues can reach.

Figure 1.2: Problem instance where the targets want to approximate to their assigned targets, while connectivity among them is kept by positioning a set of relays.



1.1 General Objectives

The general objectives of this project are:

1. Investigate and develop computationally cheap solutions for the connectivity maintenance of a set of mobile agents, named as *pursuers* trying to reach individual objectives (capture moving targets, for example) by positioning a second set of agents, named as *relays* whose mission is to form globally connected configurations over time in order to support the *pursuers* into their mission.

2. Develop an image-based tracking system in order to enable a set of drones with limited computing power to be able to track shape-shifter objects, using only one camera as input.

1.2 Thesis outline

Since this thesis is concerned about dynamic network connectivity maintenance, the Chapter 2 presents the main connectivity measures such as algebraic, edge- and vertex-connectivities. Furthermore, still in such chapter, some distance based weighting functions are also presented.

In Chapter 3, the static and dynamic connectivity problems are covered as well as their related works.

The first proposed approach, which considers a fixed network structure is depicted in Chapter 4. This first solution we propose a quadratic programming approach using a recurrent neural network as solver in order to get faster numeric results as compared to the Matlab solver `quadprog`.

In Chapter 5 it is proposed a solution based on the Laplacian Matrix of the network graph. In that chapter, we propose another connectivity function based on the estimation of the algebraic connectivity value. In that Chapter, it is also proposed a way to distribute the solution in such a way that each robot can decide where to go based only on the current neighbourhood, as well as parallel ordering of computation based on the Schedule by Edge Reversal (SER) algorithm.

In Chapter 6, some heuristic solutions are explored. We considered Genetic Algorithms, Simulated Annealing, Standard Particle Swarm Optimization 2011 and Backtracking Search Optimization algorithms.

The Chapter 7 presents the evaluation methodology as well as the numeric simulations of the proposed solutions under some test scenarios.

The Chapter 8 present a generic object tracker based on a weightless neural network. Some results are presented based on the benchmark published in [10],

where with some tuning parameters, the proposed tracker could outperform the one presented in [10].

Finally, in Chapter 9 is presented the general discussions about the developed work as well as it is pointed out some insights for future research on the discussed issues.

Chapter 2

Assessing Connectivity

The chapter 3 describes the state-of-the-art of works dealing with the connectivity maintenance. The majority of these works make use of a state-dependent network model also known as *proximity graphs*. A proximity graph has the edge set changing over time because of some weighting function that depends on the relative positions of the vertices. The current chapter aims to present some graph-based indexes of connectivity.

More precisely, in section 2.1, the edge-connectivity and vertex-connectivity are visited in order to show the basic graph-theoretic notions about connectivity. As the number of connected components is also a connectivity measure, the section **??** describes the fundamental depth-first search algorithm (also known as Trémaux's algorithm). As related by section 2.1.1, the graph connectivity can also be captured by using sums of powers of the adjacency matrix.

Beyond the previously related connectivity measures, there is a result from algebraic graph theory: the algebraic connectivity measure. Fiedler proved that the second smallest eigenvalue of the Laplacian matrix of a graph is related with its connectivity. This important measure is related in the section 2.1.2. Finally, the section 2.2 describes some of the main weighting functions used to model the state-dependent networks of the main results in the literature.

2.1 Graph-Theoretic Connectivity

Let $G = (V, E)$ be a graph with the node set V and the edge set E. The adjacency matrix of G, defined as $A = [a_{ij}]$, has its elements defined as

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

If $U \subset V$ is a vertex set for which the graph obtained by $G - U$ has more

connected components than G, then the subset U is called a vertex-cut. The vertex connectivity number $k(G)$ of a connected graph G is the minimum number of vertices whose removal would either disconnect G or reduce G to the trivial graph [20]. The edge-connectivity, denoted by $\lambda(G)$ is similarly defined. When a connected graph has the edge-connectivity of one, each edge that disconnects the graph is said *cut-edge*. A graph that does not contain such type of edges is called *bridgeless*. A graph G is said *k-edge-connected* or *k-vertex-connected* if $\lambda(G) \geq k$ or $k(G) \geq k$, respectively.

The computation of the edge-connectivity number and vertex-connectivity number of graphs has been studied and developed over the years. According to [21], most of these algorithms are based on calling a number of *max-flow* problems [22]. Because of that, attempts have been made to minimize the number of these calls. One of the first algorithms that used this approach is reported by [23] and described in algorithm 2.1.1.

Algorithm 2.1.1 Even and Tarjan's algorithm for computing the edge-connectivity between a pair of vertices (u, v).

1: **procedure** EDGE-CONNECTIVITY1$(G, (v, w))$
2: INPUT: A connected graph G and two vertices $v, w \in G$
3: Replace the edges of G with weighted arcs (weight=1 for each direction indicating the edge capacity)
4: Find a max-flow function f on this new graph
5: $\lambda(v, w) \leftarrow total\_flow(f)$
6: RETURN: $\lambda(v, w)$
7: **end procedure**

The algorithm 2.1.1 is able to calculate the edge-connectivity λ of any two vertices $u, v \in G$. In order to obtain $\lambda(G)$, it is necessary to call the procedure described by algorithm 2.1.1 for each pair of nodes in G. The minimum value should be assigned to the $\lambda(G)$. Considering a graph of n vertices, there will be $n(n-1)/2$ unordered $\lambda(u, v)$ to compute $\lambda(G)$[21]. Figure 2.1 shows some example values of edge-connectivity.

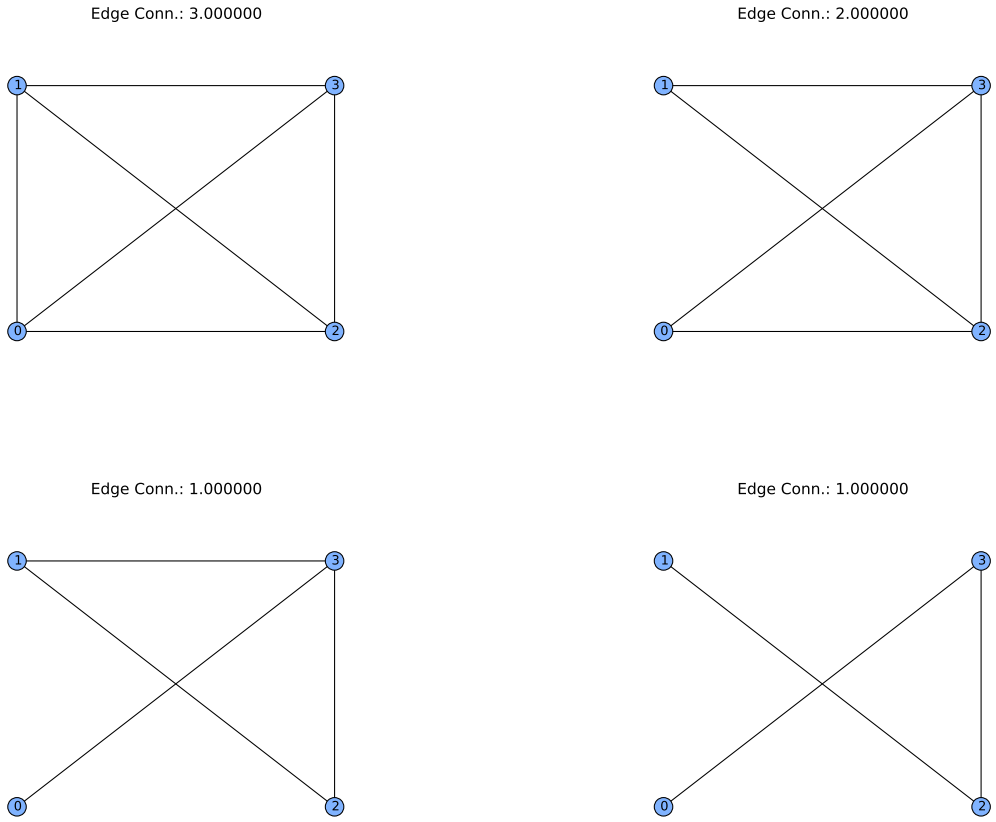


Figure 2.1: An illustration of the edge-connectivity index for different graphs.

The *minimum vertex-cut* of a graph $G = (V, E)$ is the minimum subset $S \subset V$ such that $G - S$ is either disconnected or the trivial graph. The calculation of the vertex-connectivity of G, defined as $k(G) = |S|$, can be reduced to solving a number of max-flow problems as well. Let v and w be two vertices from V, the vertex connectivity $k(v, w)$ is defined as being the least number of vertices, chosen from $V - \{v, w\}$, whose deletion from G would destroy every path between v and w, whether $(v, w) \notin E$. And in the case that $(v, w) \in E$, $k(v, w) = n - 1$. Therefore, $k(G)$ can be defined as $k(G) = \min_{v,w \in G}\{k(v, w)\}$.

When $(v, w) \notin E$, it has been shown by [24] that $k(v, w)$ can be determined by solving a max-flow problem in a particular network. This procedure is described in the algorithm 2.1.2 [21]. Figure 2.2 shows some example values of vertex-connectivity.

Algorithm 2.1.2 Even's algorithm for computing the vertex-connectivity between a pair of vertices (u, v).

procedure VERTEX-CONNECTIVITY1$(G = (V, E), (v, w))$
    INPUT: A connected graph G and two vertices $v, w \in G$
    Build a new graph $D = (DV, DE)$, such that $DV = V$ and $\forall (x, y) \in E$, add the arcs (x, y) and (y, x) in DE
    For each vertex $u \in D - \{v, w\}$, replace u with two new vertices u_1 and u_2. Add an arc (u_1, u_2). Connect all the arcs were coming to $u \in G$ to u_1, and all the arcs that were going out of u to u_2.
    Assign a *capacity* $= 1$ to all the arcs of D.
    Assign v as the source vertex and w as the sink vertex.
    Find a max-flow function f on the resulting graph.
    $k(v, w) \leftarrow total\_flow(f)$
    RETURN: $k(v, w)$
end procedure

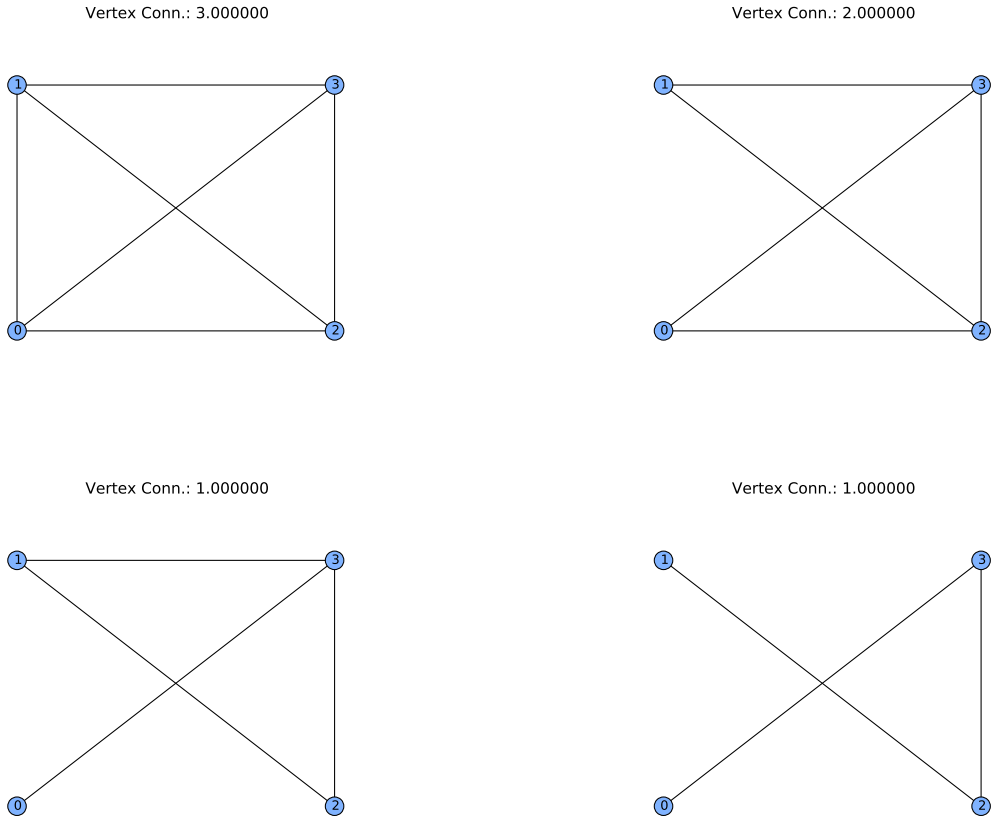


Figure 2.2: An illustration of the vertex-connectivity index for different graphs.

According to [21], the time complexity of the algorithm 2.1.2 is $O(mn^{2/3})$. The literature review of graph algorithms for calculating the edge- and vertex-connectivity include improvements on the number of calls on the max-flow

procedures. In addition, other techniques have been tested that vary from the use of depth first search [25] and dominating sets[26] to randomised techniques [27]. In [21] the author shows a table with a chronologically ordered list of connectivity algorithms.

Another surrogate approach to get the vertex connectivity is using a search based algorithm. According to [24], the pioneer work of Trémaux consisted the first algorithm for doing a Depth-First Search (DFS). In such algorithm, the goal is to visit each vertex of a graph by using each edge only once (one visit going in and one visit going back). The functionality of such algorithm is as follows: starting from any vertex s, the algorithm labels any edge e with an 'E', and the other side of e is labelled with an 'F'. After all edges of a vertex is marked, the algorithm returns to the origin vertex of each call, until there is no more unlabelled edges. At the end, the algorithm returns to the start vertex s. The Trémaux's algorithm is described in Algorithm 2.1.3.

Algorithm 2.1.3 Trémaux's algorithm.

```
1:  procedure TRÉMAUX(G = (V, E), s)
2:      v ← s
3:      while There is an unmarked edge in v or v has a passage marked F do
4:          if There is an unmarked edge e = (v, u) then
5:              Mark the edge e at b as E
6:              if u has no marked passages then
7:                  Mark e at u by F
8:                  v ← u
9:              else
10:                 Mark e at u as E
11:             end if
12:         else                                  ▷ There is an edge in v marked as F
13:             Use the edge marked F to move to the neighboring vertex u
14:             v ← u
15:         end if
16:     end while
17: end procedure
```

Then, whether the number of vertices is known, the Trémaux procedure allows to check for the connectivity of the graph, by returning the number of visited vertices.

2.1.1 Connectivity by powers

According to [28], the powers of the adjacency matrix can reveal if a graph is connected or not. The entry $[A^k(t)]_{ij}$ of the matrix is the number of paths of length k from i to j. Then, if there is an integer K such that all entries of the matrix $C_K = \sum_{k=0}^{K} A^k$ are greater than zero then the graph is connected.

It is important to say that the integer K is upper-bounded by $n-1$, where n is the number of vertices on the graph. In [29], Zavlanos *et. al.* state that whenever the network begins in a connected setting, if $K = 1$ it is applied (keep one-hop neighbors) of an originally connected network, then the network will remain connected for all time. The Figure 2.3 shows some examples of this index. As can be seen on the picture, the value of power connectivity increases when the longest path between any two vertices becomes longer (in terms of edge counting).

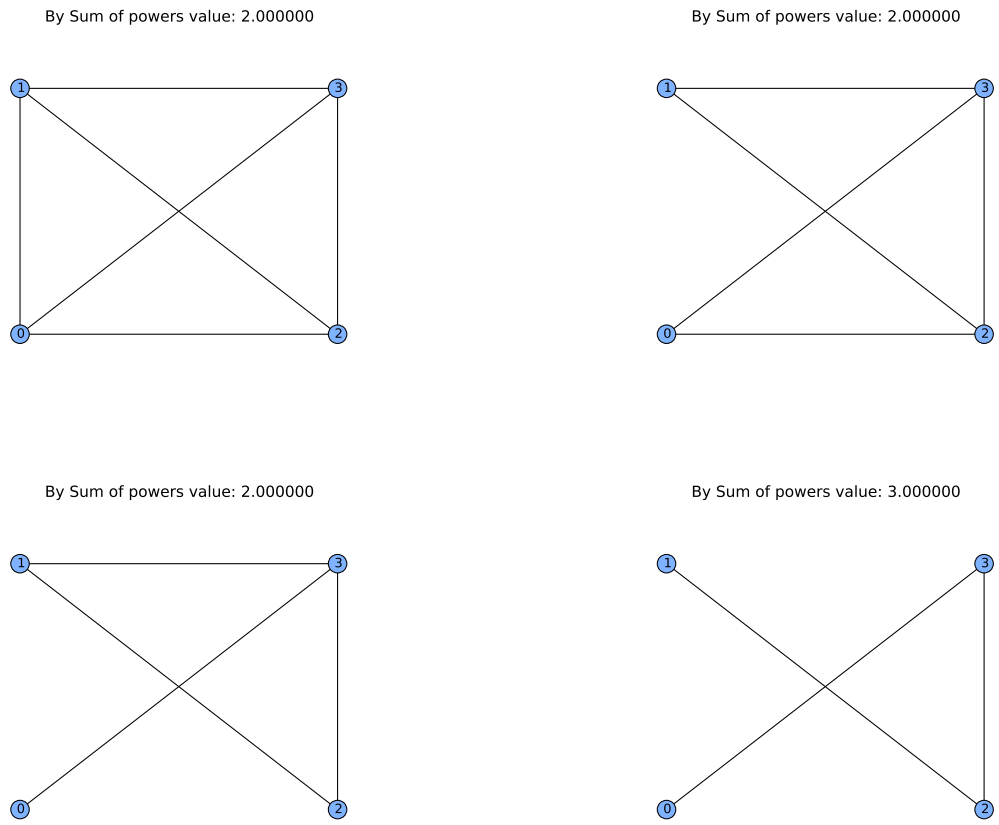


Figure 2.3: An illustration of the connectivity by sum of powers for different graphs.

2.1.2 Connectivity by Spectral Graph Theory

The spectral graph theory is concerned of studying the spectral properties of the matrix representation of a graph. From the adjacency matrix, the Laplacian matrix can be defined as follows: Let W be the weighting matrix of a graph G. Let $D = [d_{ij}]_{n\times n}$, with $d_{ij} \in \mathbb{R}$, a diagonal matrix, defined by $d_{jj} = \sum_{i=1}^{n} w_{ij}$ (node degrees), with $j = 1, \ldots, n$. The Laplacian matrix can be defined as $L = D - W$.

According to [30–34] the following properties holds for a Laplacian matrix of a graph G with non-negative and symmetric weights:

Proposition 1. *The matrix $L(G)$ (simply L for short) with non-negative and symmetric weights has the following properties:*

(i) For each vector $x \in \mathbb{R}^n$, $x^T Lx = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2$.

(ii) L is a symmetric semi-definite positive matrix.

(iii) L has n non-negative eigenvalues (graph multiplicity).

(iv) The least eigenvalue of L is equal to 0, with the corresponding eigenvector $\alpha\mathbf{1}$, $\alpha \in \mathbb{R}$, and $\mathbf{1}$ the unitary vector with dimension n.

As presented in [34], to prove *(i)*, consider the following inequalities:

$$x^T Lx = x^T Dx - x^T Wx = \sum_i d_{ii}x_i^2 - \sum_{i,j} w_{ij}x_i x_j$$

By the above, it is concluded that L is a positive semi-definite matrix ($x^T Lx \geq 0$, since all weights are non-negatives). By definition, $L = D - W = D^T - W^T$ which proves the property (ii). Property (iii) is a definition for positive semi-definite matrices. To prove (iv), first consider the following equation:

$$\begin{bmatrix} \sum_{i\neq 1}^{n} w_{i1} & -w_{12} & \dots & -w_{1n} \\ -w_{21} & \sum_{i\neq 2}^{n} w_{i2} & \dots & -w_{1n} \\ \vdots & & \ddots & \vdots \\ -w_{n1} & -w_{n2} & \dots & \sum_{i\neq 2}^{n} w_{in} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{i\neq 1}^{n} w_{i1} - \sum_{i\neq 1}^{n} w_{i1} \\ \sum_{i\neq 2}^{n} w_{i2} - \sum_{i\neq 2}^{n} w_{i2} \\ \vdots \\ \sum_{i\neq 2}^{n} w_{in} - \sum_{i\neq 2}^{n} w_{in} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

This means that $L\alpha\mathbf{1} = 0\alpha\mathbf{1}$, so 0 is eigenvalue of L.

The Laplacian matrix of a network G with symmetric weights is always a symmetric positive semi-definite matrix.

The mathematician Fiedler [35] proved the important result announced in theorem 1, as follows.

Theorem 1. *Let G be a graph with n vertices and L its Laplacian matrix with non-negative and symmetric weights. Let $0 \leq \lambda_1(L(G)), \lambda_2(L(G)), \dots, \lambda_n(L(G))$ be the ordered eigenvalues of L. that a graph G is connected then $\lambda_1(L(G)) = 0$ with the corresponding eigenvector $\mathbf{1}$ and $\lambda_2(L(G)) > 0$.*

The eigenvalue $\lambda_2(L(G))$, is known as the algebraic connectivity or Fiedler value of the network. Each eigenvector associated with $a(G)$ is known as Fiedler vector. The eigenvalues of $L(G)$ is also a measure for how much the graph is disconnected. For a disconnected graph G, the number of eigenvalues equals to zero is equivalent to the number of connected components of G [36]. The algebraic ($\lambda_2(L(G))$), edge

($\lambda(G)$) and vertex($k(G)$) connectivities are related, according to Theorem 2, proved in [35].

Theorem 2. *If G is not a complete graph then $a(G) \leq k(G) \leq \lambda(G)$. For K_n, $a(K_n) = n$.*

It is also used to capture the notion of k-connectivity (it is upper bounded by n).

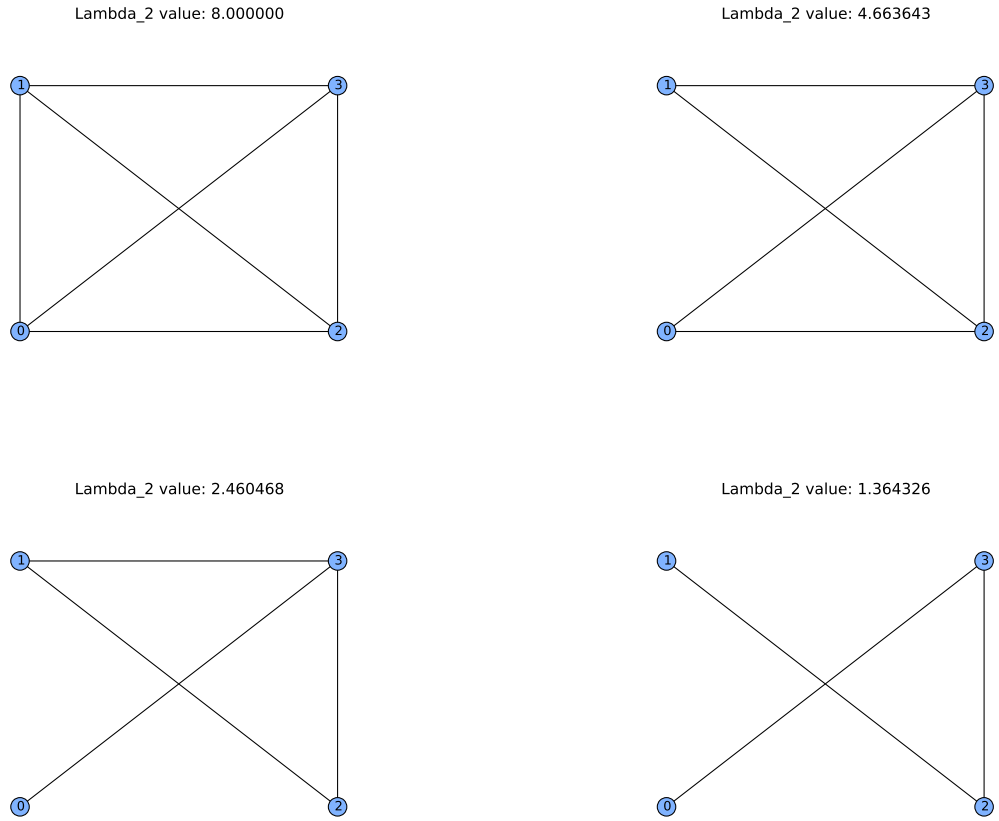


Figure 2.4: An illustration of the algebraic connectivity $\lambda_2(L(G))$ for different graphs.

Since the $L(G)$ is a Hermitian matrix, for non-zero x by the Rayleigh quotient[1], the $\lambda_2(L(G))$ takes the following form [37, 38]:

$$\lambda_2(L(G)) = \min_{x \perp \mathbf{1}, x \neq \mathbf{0}} \frac{x^T L x}{x^T x} = \min_{x \perp \mathbf{1}, x \neq \mathbf{0}} \frac{\sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2}{||x||^2} \tag{2.1}$$

Another important result using the Laplacian Matrix is the Matrix-tree theorem, associated to Kirchhoff, because of his studies in electrical circuits for which he

[1]Special thanks to professor C. Justel for this contribution.

proved that the number of spanning trees is given by any cofactor of L. As announced in Theorem 3.

Theorem 3. *[36] Matrix-tree Theorem*

$$adj(L) = \tau(G).J$$

where $\tau(G)$ is the number of spanning trees of G and J is the matrix with all entries equal to 1.

2.2 Weighting Functions

When dealing with coordination for multi-robot systems, the connectivity is seen as a distance function. Positional solutions consider the Euclidean distance and position the robots accordingly to this distance. Although it is straightforward to deal with that, some solutions though employ some other weighting functions to deal with more realistic radio scenarios. This section describes some of these weighting functions as well as some ratios used by some solutions.

A proximity graph is defined as $G(t) = (V, E(t))$ for which the edge set is dependent on the relative state of the elements of V. The adjacency matrix of $G(t)$, defined as $A(t)$, has its elements defined as $[A(t)]_{ij} = w_{ij}$. Where, w_{ij} is a function of the relative state of V.

In the case of dealing with electronic chaining algorithms, some authors would rather prefer to use more realistic ratios. Considering for example the Radio Frequency physical space, other than purely distance based functions. During this research, two of these ratios have been noticed: the Signal-to-Noise (SNR) and the Signal-to-Interference and Noise Ratio (SINR).

The Signal-to-Noise, as noted by [39], between two nodes i and j is defined as:

$$SNR_{ij} = \frac{P_{ij}}{No_i} \tag{2.2}$$

where P_{ij} the the power received by node i from the transmission node j, and No_i is the environmental noise seen by node i.

Used as the signal model by the solution in [40], the Signal-to-Interference and Noise Ratio (SINR) is also another metric which the link between two agents i and j increases with the improving of communication strength, as well as, it decreases as the environmental noise (No_i) around node i increases and the interfering communication amongst i's other neighbors, as seen from the definition in (2.3)

[41]:

$$SINR_{ij} = \frac{f_{ij}}{No_i + \sum_{k \in N_{i \setminus j}} f_{ik}} \tag{2.3}$$

where $N_{i \setminus j}$ is the set of neighbors of i that are different from j, f_{ij} is the communication strength over the link $i - j$.

Besides, there are other weighting functions that are based on the Euclidean distance, and when used in spaces free of any obstacles, decay somehow with the increase of the distance. To illustrate, the authors of [29, 37, 38], propose the utilization of weighting functions as the one shown in Figure 2.6. This one in particular is well accepted because it seems to fit at the representation of the wireless signal. In one hand, once a robot is close enough to a neighbor ($\leq \rho$), the signal strength could reach its maximum value. By the other hand, as the robots increase their distance between each other, the signal decay exponentially until reach the minimum threshold to consider the connected state link as disconnected.

In addition to the weighting function presented in Figure 2.6, the authors of [29, 38] also present other weighting functions of the inter-robot distance, as can be seen in (2.4), (2.6) and (2.7) and with their respective plots in Figures 2.5, 2.7 and 2.8.

$$f_a(d) = \begin{cases} 1 & \text{if } d \leq \rho_2 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

where d is the distance between any pair of nodes and ρ_2 is the communication threshold.

$$f_b(d) = (1/(1 + e^{-\alpha(d-\rho)})) \tag{2.5}$$

with $\alpha = (2/(\rho_2 - \rho_1)) \log((1 - \epsilon)/\epsilon)$, and $\rho = (\rho_1 + \rho_2)/2$

$$f_c(d) = \begin{cases} (1/(\rho_1 - \rho_2))/d - (\rho_2(\rho_1 - \rho_2)) & \text{if } \rho_1 \leq d < \rho_2 \\ 1.0 & \text{if } d < \rho_1 \end{cases} \tag{2.6}$$

$$f_d(d) = e^{-\alpha(d-\rho)} \tag{2.7}$$

with $\alpha = (1/(\rho_2 - \rho_1)) \log(1/\epsilon)$ where d is the distance between any pair of nodes and ρ_1 is the maximum distance which gives the maximum communication capacity, $\rho_1 < \rho_2 < d_{max}$ is the decay threshold and d_{max} is the distance with minimum communication.

Now that the necessary background on the graph theoretical related concepts altogether with the notions of weighting functions, the description of the related works follows in the next chapter.

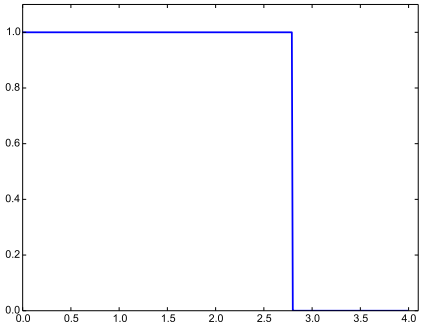


Figure 2.5: (a) A plot of the weighting function defined in (2.4).

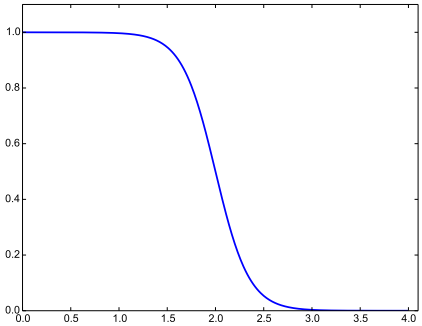


Figure 2.6: (b) Plot of the weighting function in (2.5).

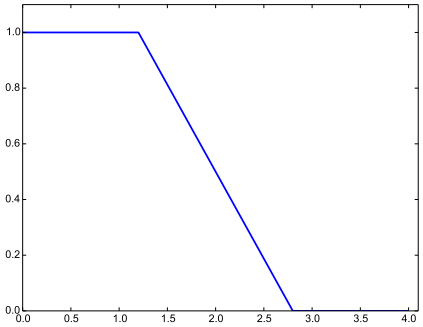


Figure 2.7: (c) A plot of the weighting function defined in (2.6).

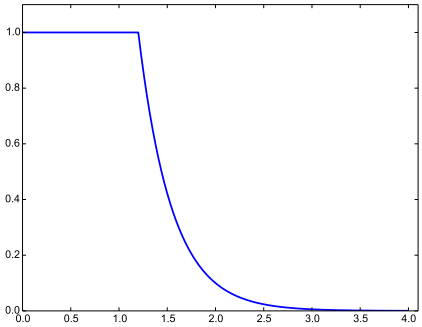


Figure 2.8: (d) A plot of the weighting function defined in (2.7).

Chapter 3

Related Works: Dealing with communication constraints

In this chapter, it is first discussed the static connectivity problem, its correlation with the Steiner tree problem, as well as its complexity (Section 3.1). Furthermore, the connectivity problem is investigated through the description of some works in two separate ways. In Subsection 3.2.1 some relay chain approaches are shown, presenting the complexities of dealing with the stabilization of connectivity between two end points. Moreover, on Section 3.2.2, the approaches for the multi-end points connectivity problem are described in order to present some works and their proposed solutions.

Notation

- T target set or terminal points. This set can be indexed by time.
- P set of active agents. This set is usually defined when there is situations where the agents can be split in two or more objectives.
- R set of relay agents. The mission of the elements of such set is usually maintain the connectivity or deal with it.
- R_{comm} it is the communication threshold. If any pair of agents is positioned such that the distance between them is more the R_{comm}, it is said that there is no communication between them.
- $\mathbb{R}^{n\times n}$ space of $n \times n$ matrices.
- $\mathbb{S}^n$ space of real symmetric $n \times n$ matrices.
- I_n is the identity matrix of order n.

- $A \succ B$ and $A \succeq B$ indicate the positive definiteness and positive semidefiniteness of the matrix difference $A - B$ (A and B being symmetric matrices[1]).

3.1 The static connectivity problem

This thesis deals with the problem of dynamic positioning a set of agents under connectivity constraints. But before relating the literature review over this theme, it is important to present a branch of the static scenario, which looks for the positioning while the number of employed extra points is also minimized.

Considering the static scenario, where given a set T (with $|T| > 2$) of stationary targets positioned in an Euclidean Space, and a constant $R_{comm} > 0$ indicating the communication radius, the *placement problem* consists on finding a set of extra points C and their corresponding positions such that $|C|$ is minimized and the proximity graph generated by the position of such points is connected.

In wireless sensor literature, the placement problem is classified according to two configurations: *one-tier*, when the relay connection between any pair of sensors is allowed; and the *two-tier* version that restricts the connection to happen only between terminal-to-relay and relay-to-relay (terminal means terminal points).

In [1], the authors present an Incremental Optimization Delaunay Tree algorithm, called *IO-DT*, which optimally solves the SMT-MSPBEL problem for the case of three terminals. The algorithm iterates over the triangles given by the Delaunay Triangulation solving the problem for each triangle and incrementally composing the global solution. An example of the Delaunay Triangulation applied to 10 random points can be seen in Figure 3.1

The following five definitions would help to understand the further paragraphs ([1]):

Definition 1. *The minimum number of Steiner Points (SPs) required for forming a SMT-MSPBEL for two terminals u and v is called the* ***sp-weight*** *and defined as*

$$W_{sp} = \left\lceil \frac{||d(u,v)||}{r} \right\rceil - 1$$

where r is the communication radius.

Definition 2. *Let T_i be a triangle with vertices (u, v, w). The* ***Fermat point (FP)*** *f_i for T_i is the point that minimizes the sum of the distances from f_i to all vertices in T_i. Given an integer positive constant r, the* ***Discrete Fermat Point*** *of a triangle $T(u, v, w)$ is a point x such that minimizes*

[1] Löwner ordering

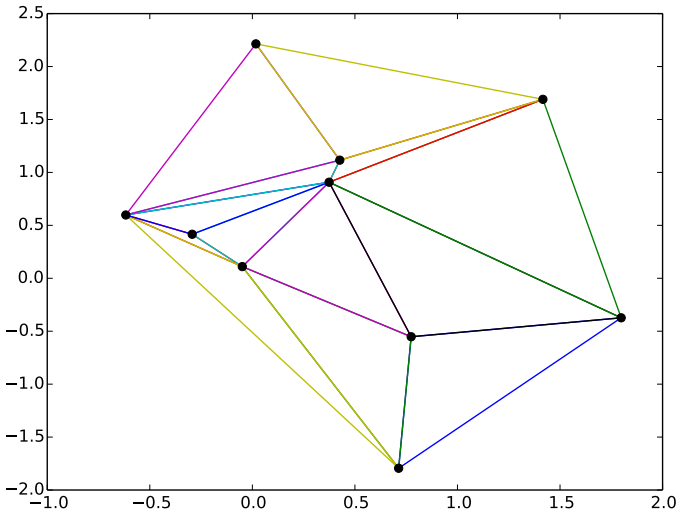


Figure 3.1: Example of the Delaunay Triangulation application over 10 points.

$$\left\lceil \frac{||ux||}{r} \right\rceil + \left\lceil \frac{||vx||}{r} \right\rceil + \left\lceil \frac{||wx||}{r} \right\rceil$$

Theorem 4. *The number of points, which can possibly be DFP for a $T(u,v,w)$ does not exceed $\lceil 4s(s-|uv|)(s-|uw|)(s-|vw|)/r^2 \times |uv|+|uw|+|vu|/vw \times |uw| \times |uv| \rceil$ where s is the semi-perimeter of T.*

Definition 3. *Let T_i be a triangle with vertices (u,v,w) and x a point inside T_i. The number of relays required for connecting the vertices of T_i, called **fp-weight**, is denoted by:*

$$W_{fp}(T_i,x) = (\left\lceil \frac{||ux||}{r} \right\rceil - 1) + (\left\lceil \frac{||vx||}{r} \right\rceil - 1) + (\left\lceil \frac{||wx||}{r} \right\rceil - 1) + 1$$

Definition 4. *Let T_i with vertices (u,v,w). Given a minimum spanning tree G_{mst} that includes u,v, and w, assume that Y_1 and Y_2 are two paths in G_{mst} from u to v and from u to w, respectively. Let e_1 and e_2 be the edges having maximum **sp-weight** in Y_1 and Y_2, respectively. The **mst-weight** of T_i is denoted and computed as follows:*

$$W_{mst}(T_i) = \begin{cases} W_{sp}(e_1) + W_{sp}(e_2) & if e_1 \neq e_2 \\ W_{sp}(e_1) + max\{W_{sp}(e_3), W_{sp}(e_4)\} & otherwise \end{cases}$$

*Note that if a triangle T_i contains two mst edges $e_1 = (u,v)$ and $e_2 = (u,w)$ then the **mst-weight** of T_i will be the sum of $W_{sp}(e_1)$ and $W_{sp}(e_2)$.*

Definition 5. *The **min-weight** of a triangle T_i is the minimum of **mst-weight** and **dfp-weight** of T_i*

$$W(T_i) = \min W_{sp}(T_i), W_{dfp}(T_i)$$

The ***gain*** *of* T_i *is defined by* $Gain(T_i) = W_{mst}(T_i) - W_{dfp}(T_i)$

The first case treated by [1] is the 3 points case. They define the Discrete Fermat Point (DFP) optimization as the procedure of computing the candidate point of minimum fp-weight. This candidate is elected among a set of intersection points inside the triangle. These intersection points are obtained after drawing circles of radii $r_i = \{r, 2r, \ldots, kr\}$, where $r = R_{comm}$ and centered at the vertices of the triangle. The Figure 3.2 illustrates their idea. In order to avoid triangles with terminal points inside it, they utilize the Delaunay Triangulation. As a result, the IO-DT outperformed the other heuristics, employing fewer relays and generating nodes with less node degree and expected path length.

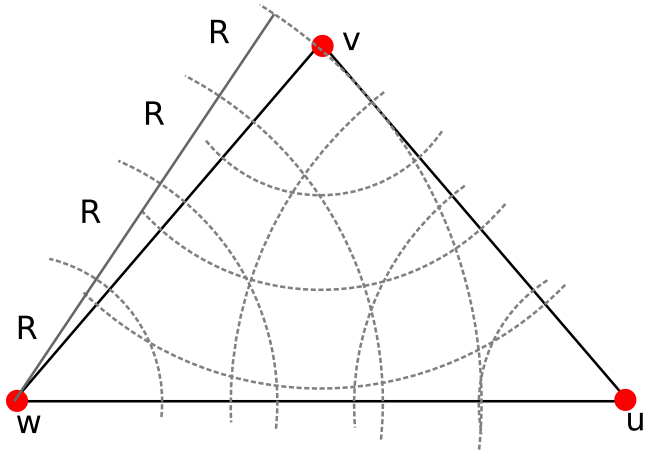


Figure 3.2: The intersection points are Steiner candidates [1]. R indicates the R_{comm}.

Despite the approximation factor of IO-DT had not been calculated, the authors showed that its complexity is $O(n^2)$, where $n = |T|$ is the number of terminals. In order to evaluate the performance of their heuristic, they validated IO-DT through simulation in a scenario varying the number of terminals (10 to 25) deployed in random positions inside an area of $1500m^2$, with $R_{comm} = 100m$. The authors compare their heuristic with the 3-approximate proposed in [42], SMST [43, 44], and FeSTA (also proposed by the same author) [45].

When $|T| > 3$ (and do not form a straight line) the version of the placement problem takes a harder degree of complexity. Cheng et al. [42] modelled the placement problem as an instance of the Steiner tree problem with minimum number of Steiner points and bounded edge-length (STP-MSPBEL for short) [43].

The STP-MSPBEL was proved to be NP-complete by [43] through a reduction of an instance of STP-MSPBEL to an instance of the Euclidean Steiner Minimal Tree (ESMT) [46], that is proved to be at least as difficult as any of the NP-complete

problems. The reduction used by [46] consisted of transforming a discrete version of the ESMT (the superior integer value of the Euclidean distance) in to an instance of the 3-Set cover instance, a well-known NP-complete problem.

To understand a summary of his finding, we borrow the following two definitions from the work in [43]:

Definition 6. *DESMT - Discrete Euclidean Steiner minimum tree is defined over a given a set of terminals T of integer-coordinate points in the Euclidean plane, the decision problem is defined as given a positive integer L, does there exist a superset $C \supseteq T$ of integer-coordinate points such that some spanning tree G_{tree} of Y satisfies $l'(G_{tree}) \leq L$? (where $l'(T)$ is the discrete length of tree G_{tree}).*

Definition 7. *STP-MSPBEL. Given a set T of m terminal points in the two-dimensional Euclidean plane, a positive constant R_{comm}, and a non-negative integer L. The problem asks whether there exists a tree spanning a point set $Q \supseteq P$ such that each edge in the tree has a length no greater than R_{comm} and the number of Steiner points ($C \supseteq T$) is less than or equal to L.*

Theorem 5. *[43] There is a polynomial time reduction from the DESMT to STP-MSPBEL.*

After defining the complexity of the correlated placement problem, the remaining of the text is concentrated with the survey on works that deal with some variants of this problem. Furthermore, in [47] the authors approached the problem of finding Pareto-optimal chains that include minimizing the number of relays and maximizing the chain connectivity quality. They considered a scenario with obstacles which the objective consisted of finding the positions of relays in order to connect a fixed base station to a fixed target for surveillance purposes. This scenario is illustrated on Figure 3.3. In their approach, the environment is firstly preprocessed in order to obtain a discretized version of it. Only points where the UAVs could safely navigate and outside of the obstacle areas were considered. For each node in the resulting graph, the preprocessing step also includes the creation of oriented edges to reachable nodes, with associated weights that correspond to the cost of communication to those nodes.

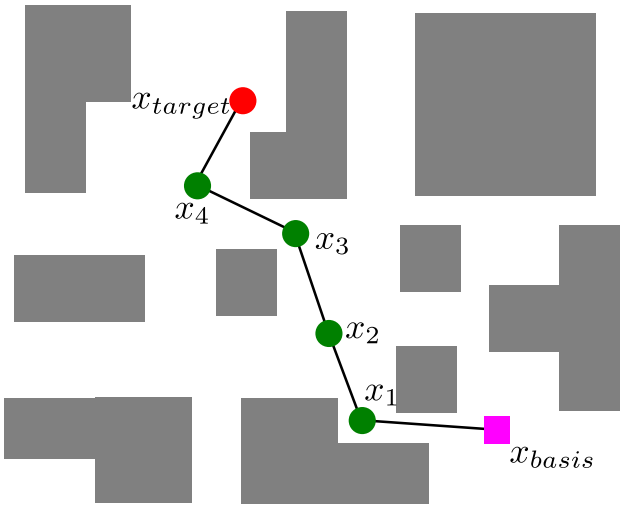


Figure 3.3: Vehicles positioned at $x_1, \ldots, x_4$ are acting as relays in order to connect a base station ($t_0 = x_{basis}$) to enable surveillance to a target located at $t_1 = x_{target}$[2].

Olsson *et. al.* [2] improved the work of [47] by introducing the support for various targets. They first used a cheapest path heuristic to find a first version of the relay tree and thereafter they applied local searches to improve the obtained tree. The cheapest path heuristic works by initializing the tree with the cheapest path from the base station to one of the targets and by adding the other cheapest paths from each target to a non-target node already in the tree.

In order to optimize the relay trees, the authors proposed different optimization criteria, such as generating the least cost tree, the tree using the least number of relays or the least cost tree using at most a fixed number of robots. They proceeded as if the current tree is infeasible, it is first optimized with regard to the number of relays. Once a feasible tree is found, the optimization criterion is changed to finding the least cost feasible tree. This process is repeated until no better improvements are found or the available time runs out. The improvements consisted of identifying nodes that has a number of connexions greater than 2, known as Steiner nodes. The feasible subtrees discarding the non Steiner nodes become substitutive candidates. Current subtrees with higher costs than the candidates are replaced.

In [48] , the authors considered a scenario with obstacles where there is a gateway (it can be seen as a base station) and a number of source spots that should be connected using a minimum number of relay robots. The authors developed two types of algorithms: one based in the optimal link measure and another one based on the approximation of the Steiner tree. In both algorithms, the objective was the selection of the minimum number of randomly deployed robots to compose a backbone connecting the sources to the gateway.

3.2 Using mobility to reach connectivity

In the previous Section de hardness of the corresponding problem of finding the minimum number of extra points with limitations on the radius has been described. Besides, some works that tries to find the positioning of such points, while maximizes the signal or minimizes the number of employed points were described. Since this thesis is concerned with connectivity maintenance using mobile robots, the related works is described into two groups, the ones that employs relay chains (section 3.2.1) and the ones that considers the connectivity among more than two endpoints (Section 3.2.2).

3.2.1 Relay chain approaches

The simpler scenario considered by this thesis consists of giving connectivity to a set of two points. This Section describes some works concerning this basic scenario. Considering the scheme where the two static points s and t, and a set of mobile robots are deployed over the area that contains s and t, in [3], the authors considered two different branches for the problem: a) a budget B of the total amount of traveled motion ($sumDist$); and b) an equal limited amount of movement for each robot ($maxDist$). The authors announced this problem as the problem of building communication bridges with the objective of minimizing the number of hubs, while simultaneously minimizing the robots' motion. Figure 3.4 illustrate an instance of this problem.

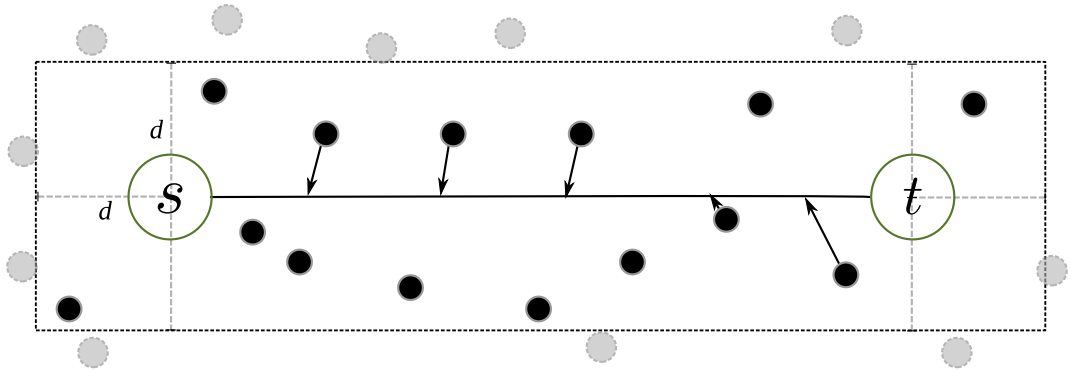


Figure 3.4: An illustration of an instance problem dealt in [3].

As pointed out in [3], one of the major drawbacks of this problem is the lack of an "ordering property" in the optimal solution. By considering the version where there is a limitation on the travelled distance by each robot, the Figure 3.5 illustrates an example of this issue. Suppose d is the maximum travelled distance allowed for a robot and r is the communication radius. Consider two robots a and b, located in a way that their respective intervals that lies into the segment $[s, t]$ are $[l_a, r_a]$ and

$[l_b, r_b]$. It is possible to build instances where the interval $[l_a, r_a]$ lies into $[l_b, r_b]$. Therefore, despite r_a is on the right of r_b, in the optimal solution the robot a moves to the right of the robot b.

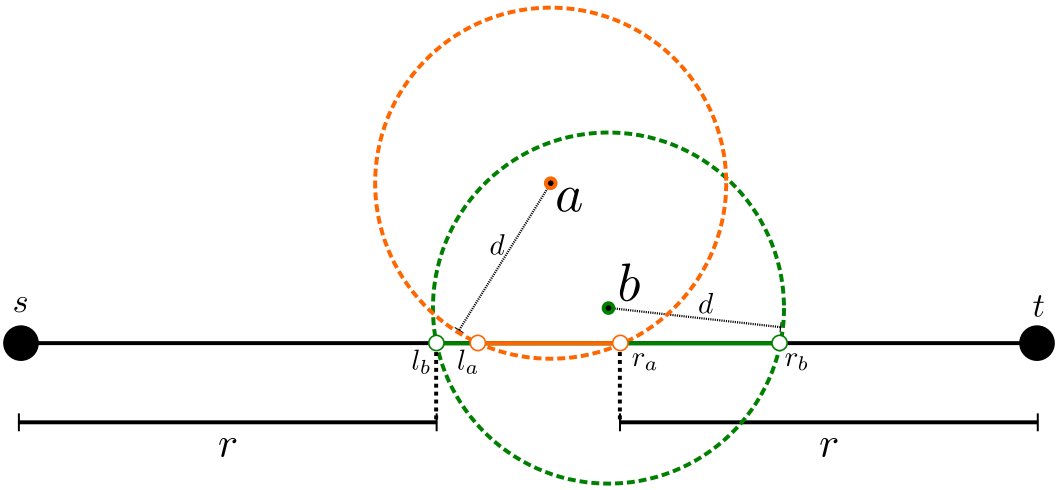


Figure 3.5: An illustration of the lack of ordering in a problem instance of the communication bridge construction problem [3].

For the *maxDist* case, the authors started by selecting a subset $P' \subseteq P$ of the deployed relays (P, in order to keep only the relays capable of reaching the line segment between s and t. This selection can be seen in Figure 3.4. For each hub $p_i \in P'$, a line segment $l_i : [x_i - d, x_i + d]$ is computed. The final location of the hub p_i is then picked from l_i. Therefore, for the final location of $x_i^{'} \in [x_i - d, x_i + d]$ the distance travelled by p_i is not greater than $\sqrt{2}d$.

A placement is said to be *well-ordered* whenever $x_i \leq x_j$ implicate $x_i^{'} \leq x_j^{'}$, for any two hubs p_i and p_j. The authors of [3] claimed that there is an ordering property for the relaxed version of the *maxDist* problem. They argued that in an optimal placement, let (p_i, p_j) be an unordered consecutive pair, for which the respective final locations at $x_i^{'}$ and $x_j^{'}$, with $x_i \leq x_j$ but $x_i^{'} > x_j^{'}$. From the relaxed segment assumption, the following items can be observed: i) $x_i^{'} \leq x_i + d$ and $x_j - d \leq x_j^{'}$ holds; ii) $x_i \leq x_j$ and $x_i^{'} > x_j^{'}$. From i) and ii), $x_i - d \leq x_j - d \leq x_j^{'} < x_i^{'} \leq x_i + d$. From the fact that $x_i - d \leq x_j^{'} < x_i^{'} \leq x_i + d$ holds, hence the robot p_i can be moved to $x_j^{'}$ which is a point that lies into the its feasible region. The same can be done with the robot p_j.

For the *sumDist* version, the authors claimed that there is an optimal solution which satisfies an ordering property, when using the L_1 metric. Using this metric, there exists a *well-ordered* optimal solution [3]. Suppose OPT_1^* is an optimal solution with the least number of unordered pairs. Let p_i and p_j be consecutive hubs in OPT_1^*, such that $x_i \leq x_j$ but $x_i^{'} \geq x_j^{'}$. Let $b = |x_i - x_i'| + |x_j - x_j'|$ and $b' = |x_i - x_j^{'}| + |x_j - x_i^{'}|$ be two budgets. Fixing the locations of x_i and x_j, it may be concluded that swapping

the positions of p_i and p_j does not increase the total budget, which can be verified by the following situations [3]:

$$cx'_j < x'_i \leq x_i \leq x_j \rightarrow b = b' \tag{3.1}$$
$$x'_j \leq x_i < x'_i \leq x_j \rightarrow b > b' \tag{3.2}$$
$$x'_j \leq x_i < x_j \leq x'_i \rightarrow b \geq b' \tag{3.3}$$
$$x_i \leq x'_j < x'_i \leq x_j \rightarrow b > b' \tag{3.4}$$
$$x_i \leq x'_j \leq x_j < x'_i \rightarrow b \geq b' \tag{3.5}$$
$$x_i \leq x_j \leq x'_j < x'_i \rightarrow b = b' \tag{3.6}$$

Let $OPT(d)$ be the number of relays in an optimal solution to *maxDist* with distance constraint d. The authors showed that the relation between the unrestricted version $OPT(d = \infty) = \lceil |st/r| \rceil - 1$ and $OPT(d)$ is less or equal to 2 (in orther words: $OPT(d)/OPT(\infty) \leq 2$).

In [4], the authors approached the problem of computing the minimum number of robotic routers (and their motion strategies) in order to maintain the connectivity of a *single user* to a *base station*. They assume an upper bound on the number of robots necessary to give connection to the user through the environment, independently the user trajectory. The authors also assume that the robots do not know the user trajectory *a priori* and the relay robots are aware of the user's position all the time. The approached problem can be defined as: given an environment $\mathcal{P}$ (possibly with obstacles) and a base station $b \in \mathcal{P}$, find the minimum number of robotic routers and their motion strategies such that wherever the user u moves, it is connected to the base station at all times, and the motion and communication constraints are satisfied.

Their results are threefold: a) an approach called EQ-DIST which maintain the Evenly Spaced Property (ESP) on the robot chain between the base station the user in a concave polygon; b) an approach that uses a robotic wrapping arm and a connection arm to handle the scenario with one obstacle in a concave polygon; and c) an approach for the convex polygon with multiple obstacles. Each approach is illustrated in Figures 3.6, 3.7, and 3.8 respectively. The main result of this work consisted of showing that if the robotic routers have the same maximum velocity capacity of the user, then it is possible to move the relay robots in order to maintain the user connected to the base station.

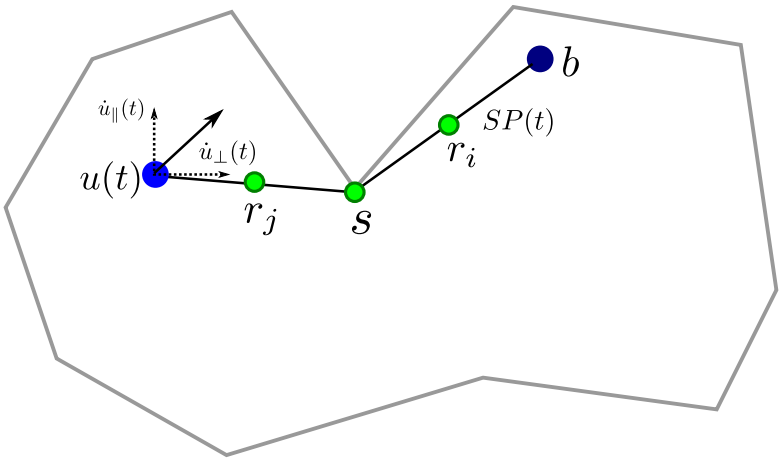


Figure 3.6: The ESP is maintained updating the tangential component of the velocity [4].

The first scenario, with no obstacles, the authors assume that the ESP holds for time (0). This means that the user must wait for the chain initiate the connection before starting to move. Calling $SP(t)$ is a polygonal chain $\{p = b, p_1, p_2, \ldots, p_j, u\}$, where b represents the basis position, u the user position, and p_i the position of the relay robot i. A *parent node* of any point z on SP is defined as the node between b and z that is the closest to z. Let $z^{\dagger}(t)$ be the parent node of a point $z(t)$. Lets parameterize the velocity $\dot{z}(t)$ into radial component $\dot{z}_{\parallel}(t)$ along the segment $[z^{\dagger}(t)z(t)]$, tangential component $\dot{z}_{\perp}(t)$. Only $u_{\parallel}$ affects the length of SP. Let $\lambda = \dot{u}_{\parallel}(t)$ be a differential change in the length of SP. So, to satisfy ESP two things must agree $\dot{r}_{i\parallel}(t) = \frac{i}{n+1}\dot{u}_{\parallel}(t)$ and if r_i lies between s and b: $\frac{|\dot{r}_{i\perp}(t)|}{|\dot{u}_{\perp}(t)|} = \frac{||sr_i(t)||}{||su(t)||}$.

The authors handle the one-obstacle situation in convex polygons by employing a wrapping arm formed by robots connecting b to a robot circle rounding the obstacle. Another connecting arm is responsible to maintain the connection between the circled obstacle and the user u. In order to control the connecting arm, the authors used EQ-DIST algorithm. Since the wrapping arm does not change over time, the authors do not employ control on it. According to the authors, such strategy uses at most $5m^*$ relay robots, where m^* is the minimum number of relay robots to give connection from the base station to the user in a convex polygon.

The last scenario dealt by the authors consider multiple obstacles in a convex polygon. In this case, they assumed the convex hulls of the obstacles are disjoint, then the polygon $\mathcal{P}$ is partitioned into cells, such that each cell is also convex and contains exactly one obstacle. For each cell, they maintain a wrapping arm that connect the obstacle to b and a connecting arm that connects u to that wrapping arm, whenever u is on that cell. The obstacles are partitioned using power diagrams. For a finite set of circles S in $\mathbb{R}^2$, the *power diagram*$(PD(S))$, is a cell complex that associates each $s \in S$ with the convex domain $\{x \in \mathbb{R}^2 | pow(x, s) < pow(x, t),\ \forall t \in$

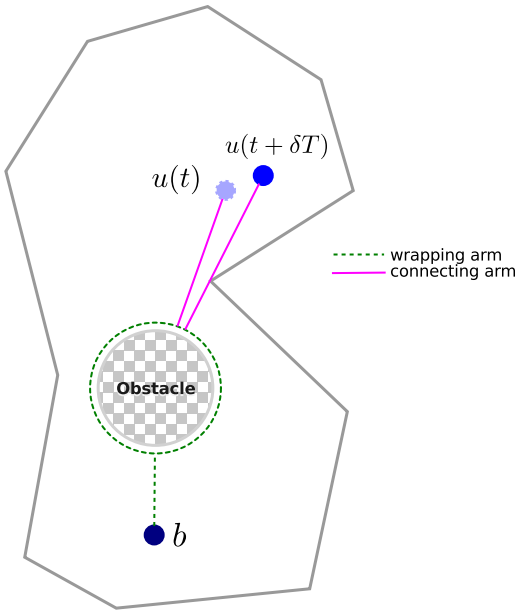


Figure 3.7: Wrapping arm to overcome the obstacle problem [4].

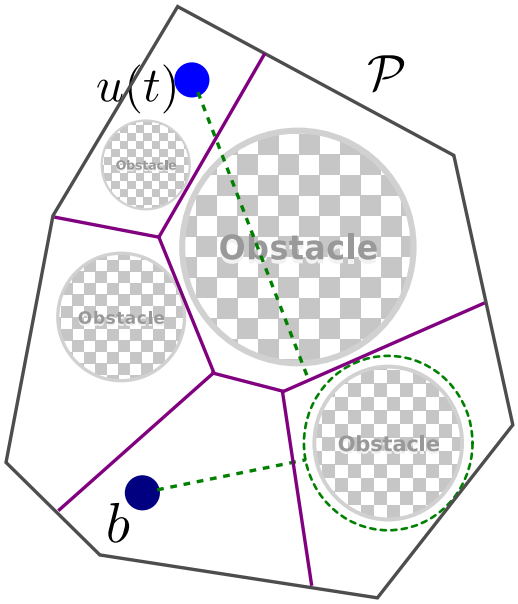


Figure 3.8: The partition $\mathcal{P}$ into power diagrams [4].

$S \setminus s\}$. Figure 3.8 shows the divided cells as well as an illustration of the strategy.

3.2.2 Connection among multiple agents

The work proposed by [49] which dealt with the problem of minimizing movement of a set of sensors for target coverage (MMTC). As shown by the authors, this problem is NP-Complete. This comes from assuming that the universal set of targets can be divided into finite subsets, it is possible to assign a weight between each pair of target and sensor. The problem become a decision problem in which is desired to minimize the weights of subsets covering all points in the universe.

Therefore, the problem become an instance of the weighted set cover problem, which is NP-complete, so as MMTC. The problem is announced as follows: given a number m of targets $T = \{t_i | i = 1, \ldots, m\}$ with known locations and n mobile sensors $R = \{r_j | j = 1, \ldots, n\}$ randomly deployed in a given task area ($n \geq m$), the MMTC problem consists in to move sensors to new positions such that all targets are covered and the total movement of relays is minimized.

The authors modeled the MMTC under the following assumptions: a) each mobile relay has the knowledge of its own position (using a localization service); b) the area has no obstacles; c) the sensors use the disk model for coverage and any target t_j is considered as covered if there is at least a relay r_j such that $dist(t_i, r_j) \leq R_{comm}$, where $dist$ is the Euclidean distance and R_{comm} is the communication radius; and d) the sensor mobility follows a free mobility model, in which sensors can move continuously in any direction and stop anywhere. The authors also assume that

there is a control center to handle the sensors locations, runs the algorithm and broadcast the resulting movements. Therefore, the connectivity among sensors has not been considered in this solution.

Despite the MMTC is NP-complete, they present a special case that arises when the distance between any pair of targets is greater than $2R_{comm}$. In such situation, the problem is equivalent to the assignment problem, that consists in assigning one sensor to each target in a way that minimizes the total cost. Then, they proposed a weighting scheme c_{ij} given by the equation 3.7.

$$c_{ij} = \begin{cases} distance(r_i, t_j) - r_s, & \text{if } dist(r_i, t_j) > R_{comm} \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

Since $n \geq m$, they added $n - m$ new columns to the weighting matrix with cost equals to 0 assigned to each new cell, in order to get an $n \times n$ matrix. Afterwards, they applied the Hungarian method [50] to solve this special case of the problem in polynomial time.

The general case has been approached through a target based Voronoi greedy algorithm (TV-Greedy). The basic idea of their algorithm consists in divide the targets space into Voronoi cells to group sensors according to their proximity. The main idea of the algorithm is divided and compacted into the following steps: 1) each sensor inside a target's Voronoi cell is called a server to this target; 2) in each target's own servers group (OSG) the closest sensor is called *chief server* and a the non-chief server is called *aid server* if it is the closest to the target's neighbours. Two targets t_j and t_i are neighbors if they share the same Voronoi vertex; 3) the target's *Candidate Server Group* (CSG) is the union of the target's own chief server with the aid server(s) from the target's neighbours.

The maximization of the global connectivity of a team of agents was considered by Kim *et. al.* in [38]. They proposed a model based on the weighted Laplacian graph for which the following weight function $\epsilon^{(\rho_1 - d_{ij})/(\rho_1 - \rho_2)}$ $\epsilon > 0$ (weighting function displayed in 2.5) was used to measure the connection between each pair of agents i, j. They proposed a discrete and greedy algorithm on a linear approximation of the problem. The problem consists in maximizing the global connectivity through the following model:

$$\max_x \lambda_2(L(G(x))) \tag{3.8}$$

where $x = [x_1, x_2, \ldots, x_m]$ are the position of the relays, $\lambda_2(\circ)$ is the second smallest eigenvalue as a function of a Laplacian matrix, and $L(G(x))$ is the Laplacian matrix of the proximity graph induced by the position of the relays. In order to prevent the agents of getting too close, they added the restriction: $dist_{ij} := ||x_i -$

$x_j||^2 \geq \rho_{safe}$, for all $i \neq j$, where ρ_{safe} is a safety threshold.

Let $L = L(G(x))$ and $\mathbf{1} = [1, 1, \ldots, 1]$ be the unitary vector whose all coordinates are equal to 1, then the authors propose that $\lambda_2(L) > 0 \equiv Q^T LQ > 0$ where $Q = [q_1, q_2, \ldots, q_{m-1}]$, and $q_i \in \mathbb{R}^m$ are unit vectors such that $q_i^T \mathbf{1} = 0, (i = 1, 2, \ldots, m-1)$ and $q_i^T q_j = 0, (i \neq j)$. So, we will have $\lambda_2(L) > 0 \equiv x^T Lx > 0$ for all non-zero $x \in \mathbf{1}^\perp$ where $\mathbf{1}^\perp = \{x \in \mathbb{R}^m | \mathbf{1}^T x = 0\}$. In this way, the model in 3.8 becomes:

$$\max_x \gamma \tag{3.9}$$

$$\text{s.t. } dist_{ij} := ||x_i - x_j||^2 \geq \rho_{safe} \tag{3.10}$$

$$Q^T L(G(x))Q \geq \gamma I_{n-1} \tag{3.11}$$

where $i = 1, 2, \ldots, m-1, j = 2, \ldots, m, i < j$, and the pairwise orthogonal unit vectors q_i's forming the columns of Q span the subspace $\mathbf{1}^\perp$.

In order to solve the model in (3.9) under restrictions (3.10) and (3.11), the authors proposed an iterative and greedy algorithm. The first step towards the solution consists in differentiate the distance function $dist_{ij} := ||x_i - x_j||^2 \geq \rho_{safe}$. The authors used the Euler first discretization method to differentiate $2\{\dot{x}_t^i - \dot{x}_t^j\}^T\{x_t^i - x_t^j\} = \dot{dist}_{ij}$. Then, $x_t = x_k$, $\dot{x}_t$ become $\frac{x_{k+1}-x_k}{\Delta t}$ producing: $2\{x_{k+1}^i - x_{k+1}^j\}^T x_k^i - x_k^j = dist_{ij}(x_{k+1}) + dist_{ij}(x_k)$. In this way, the weighting function becomes $w_{ij}(x_{k+1}) = w_{ij}(x_k) - \epsilon^{(\rho_1 - dist_{ij}(x_k))/(\rho_1 - \rho_2)} dist_{ij}(x_{k+1}) - dist_{ij}(x_k)$.

The proposed iterative step of solving the optimization model:

$$\max_{x_{k+1}} \gamma \tag{3.12}$$

$$\text{s.t.} \tag{3.13}$$

$$2\{x_{k+1}^i - x_{k+1}^j\}^T x_k^i - x_k^j = dist_{ij}(x_{k+1} + dist_{ij}(x_k) \tag{3.14}$$

$$dist_{ij}(x_{k+1}) \geq \rho_1 \tag{3.15}$$

$$Q^T L(G(x_{k+1}))Q \geq \gamma I_{n-1} \tag{3.16}$$

where $i = 1, 2, \ldots, m-1, j = 2, \ldots, m, i < j$ and $x_k := [x_k^1, x_2(k), \ldots, x_k^m]^T \in \mathbb{R}^{3m}$. Finally, they proceed to find a graph that maximizes $\lambda_2(L(G(x_{k+1})))$ until $\lambda_2(L(G(x_k)))$ can not be improved further.

Sharing the same objectives of [38], in [37], the authors applied a supergradient algorithm in conjunction with e Decentralized Orthogonal Iteration Algorithm used to compute spectral analysis (proposed in [51]). They proceeded to maximize the second smallest eigenvalue of the Laplacian matrix associated with the proximity graph represented by the network of agents. In order to get the robots navigating toward the points that maximizes the connectivity (λ_2), they used potential based

control laws that make use of only information from the agent's neighbors. With this information at hand, each agent is able to update its control law at each iteration step.

Moreover, [5] the authors considered a mixed integer linear programming approach for a pursuit evasion problem with included optional connectivity constraints. In their work, they propose a model for which the decision variables include a occupancy plan for a set of N pursuers trying to clear (cover) a given area. In their approach, the area is discretized into J cells, for which labels should be associated to them over a discretized version of time. Given an amount K of time, the model search for the set of labeling positions to N pursuers, that can move at a max speed of one cell per instant of time. The environment consisted of a 2D area with obstacles, as can be seen in Figure 3.9.

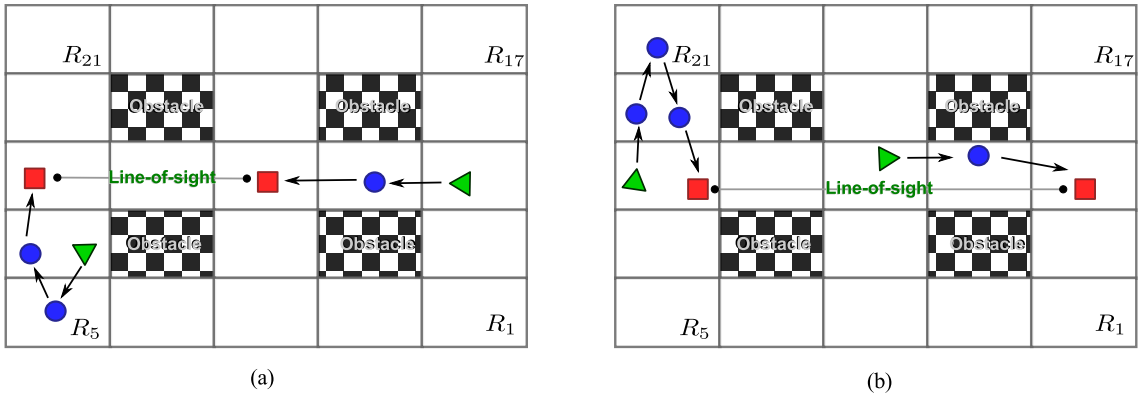


Figure 3.9: Illustration of a solution reached by the approach proposed by [5]. There are two iterations of the algorithm, for with presents the start and end of pursuers positions in order to clear the given area. Triangles, circles and squares mean start, occupancy and stop of a pursuer, respectively.

When dealing with the optional connectivity constraints, the authors of [5] considered two models: a) a general line-of-sight graph; and b) a star shaped line-of-sight graph, that the agents should be connected at a given instant of time k'. It turns out that their approach does not consider the depart point, i.e., the algorithm also decides from where the pursuers should depart of. Their approach is based on the number of cells for which the environment is divided, this means that the number of decision variables is bounded by the amount of cells used in the partition of the search area. Furthermore, there is no fixed connection limitations on the analyzed graphs, i.e., no matter the traveled distance, a pursuer can connect to another one as long as the line-of-sight condition holds. Figure 3.9 shows an example of solution for which the line-of-sight requirement is reached in the end of each iteration.

Also using discretized version of the search area, in [6] the authors approached the frontier-based exploration problem with connectivity constraints. In the frontier-based exploration problem, the frontier is defined as the collection of regions on the

boundary between open and unexplored space [52]. The authors present an extension of frontier-based exploration where the robots constantly maintain a distributed network structure, i.e, where they are constantly in contact with each other. It is based on a utility function, which weights the benefits of exploring unknown territory vs the goal of keeping communication intact. The scenario can be seen in Figure 3.10.

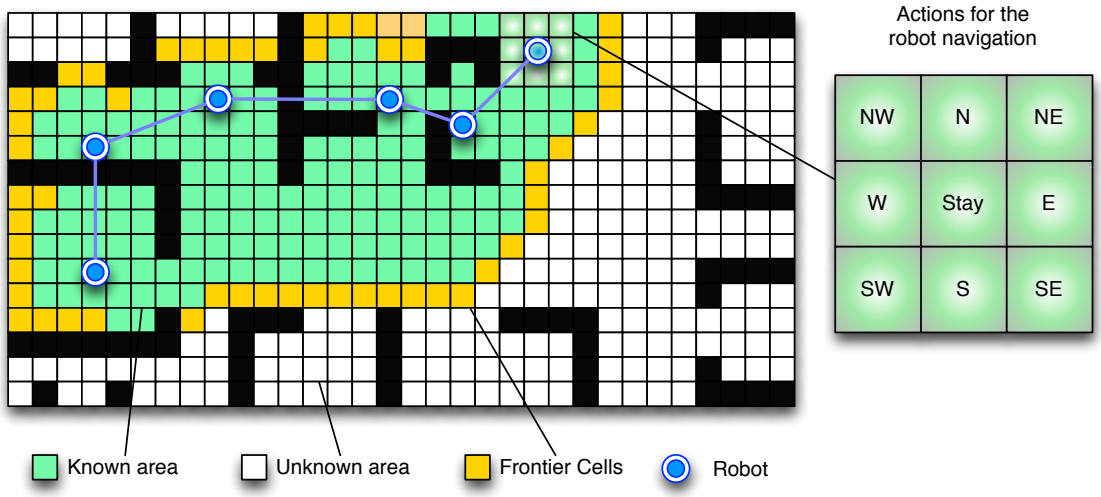


Figure 3.10: Illustration of a solution reached by the approach proposed by [6].

As can be seen in Figure 3.10, at each instant of time, the robot has at most 9 (nine) destination cells to go. Considering a team of n robots, this produces 9^n different configurations. In order to get over this problem, the authors proceeded to randomly generating a limited number of new configurations and choosing the best one, according to an utility function. Furthermore, they analysed two scenarios for the communication constraints. A first scenario considered a fixed base station where the robots should stay connected with. The second scenario considered the team of explorers to navigate through the area maintaining the connectivity among them.

The way the authors dealt with the indoor exploration leads to deadlock situations (as the one shown in Figure 3.10). The authors proposed a deadlock recover algorithm and when the robots get stuck, they break the connectivity constraint and meet at a common point. After the deadlock problem is solved, the robots come back to the exploration mode. Empirical experiments showed that a deadlock recover algorithm allowed the robots to explore the total indoor area.

Local coordination strategies for the formation problem, for which the agreement problem has been focused, was developed in [53]. They produced the control law exposed in equation 3.17, for which the agents asymptotically approach the same

point as long as the underlying graph is connected for all times.

$$\dot{x}_i = -\sum_{j \in \mathcal{N}_i} (x_i - x_j), \qquad i = 1, \ldots, N \tag{3.17}$$

where N is the number of agents, x is the position vector, and $\mathcal{N}_i$ is the set of neighbours of agent i.

In the other hand, Ji and Egersted [54] showed that when dealing with proximity graphs, the Δ-disk dynamic graph, for example, the control law in 3.17 fails in terms of maintaining the underlying graph connected. A Δ-disk dynamic graph is defined as $G(t) = (V, E(t))$ where $(v_i, v_j) = (v_j, v_i) \in E(T)$, if and only if $|x_i(t) - x_j(t)| \leq \Delta$. In order to overcome this problem, Ji and Egersted [54] proposed the control law given by equation 3.18.

$$\dot{x}_i = -\sum_{j \in \mathcal{N}_i} \frac{2\delta - ||l_{ij}(x)||}{(\delta - ||l_{ij}(x)||)^2}(x_i - x_j) \tag{3.18}$$

where x is the position vector, and $\mathcal{N}_i$ is the set of neighbours of agent i, $l_{ij} = x_i - x_j$, and δ is the communication radius. As shown in [54], the control law established in 3.18 works for a fixed topology and the initial graph should be connected. Under these conditions, the multi-agent system converges to the same point (consensus).

The authors also considered the dynamic graph case. They introduced an indicator function $\sigma(i, j)$ (equation 3.19) in order to produce a hysteresis into the process of adding edges to the underlying graph. It is important to note that there is no control of link deletions. Only adding links is allowed by this protocol. Therefore, the control law established into equation 3.18 together with the indicator function 3.19 solves the rendezvous problem under the dynamic graphs.

$$\sigma(i, j)[t + 1] = \begin{cases} 0 & \text{if } \sigma(i, j)[t] = 0 \wedge ||l_{ij}|| > \Delta - \epsilon \\ 1 & \text{otherwise} \end{cases} \tag{3.19}$$

In [55, 56], the authors designed a distributed protocol to control link additions and deletions from the network. The agents keep an estimated state of the underlying graph by means of message passing only. Using the estimation of the whole network, they control each deletion and each action of addition and deletion. The deletion and addition of links are done only once at a time and the link to be deleted or added is decided using an auction based algorithm. When the network reaches a consensus value, the involved nodes gossip the deletion information through the other neighbours in the network. Deciding on deletion of a link is only possible if the algebraic connectivity of the estimated underlying graph, kept in each agent, does not decrease to zero after the deletion of such link.

In order to add mobility to the proposed solution, the authors of [55, 56] added potential fields that blow up whenever the states tend to violate the topology established between time intervals. This means that the topology is controlled as afore mentioned, in given instants of time and in-between the agents that may navigate in a continuous way. Therefore, considering that $x_{ij} = x_i - x_j$, where x denotes the state of the agent, the authors defined the potential functions as described by equation 3.21:

$$\varphi_i = \sum_{j \in \mathcal{N}_i} \varphi_{ij} \tag{3.20}$$

$$\varphi_{ij}(x_{ij}) = \frac{1}{||x_{ij}||^2} + \frac{1}{R_{comm}^2 - ||x_{ij}||_2^2} \tag{3.21}$$

where R_{comm} is the connectivity threshold, $\mathcal{N}_i$ is the neighborhood of i.

Based on the equations 3.20 and 3.21, the authors came up with the control: $u_i(t) = -K \nabla \varphi_i(t)$, for which it is guaranteed that all links in the underlying topology are maintained and collisions are avoided. These models were tested through simulations using the Gazebo and Player [57] simulators as well as into real robots [58].

In [59], the authors evaluate the RSSI (Received Signal Strength Indicator), throughput and packet loss rate as link quality metrics while present the corresponding control schemes for the distributed control of a team of robots in order to maintain the connection among them. In their approach, the authors aim to present a solution that allows the robots to dynamically reconfigure themselves to maintain reliable communication links.

The authors used the Delaunay Triangulation scheme to build the one-hop network information. The network was represented using a matrix of adjacency A where the connectivity info between a pair of robots r_i and r_j is filled by the following scheme:

$$A_{ij} \triangleq \begin{cases} \Delta_{ij}, & ||dist_{ij}|| < \rho_1 \\ 0, & ||dist_{ij}|| > R_{comm} \\ exp[\frac{-5(||dist_{ij}|| - \rho)}{R_{comm} - \rho_1}], & otherwise \end{cases}$$

where Δ_{ij} is a measure of signal strength between i and j, for each pair (i, j) such that i is neighbor of j according to the Delaunay Triangulation, ρ_1 is a communication signal quality threshold and R_{comm} is the communication threshold . For all other robots not in the neighborhood of i, $A_{ij} = 0$.

As the Delaunay triangulation is time consuming for a multi-robot system, the

authors proposed a decentralized, potential field based method to cooperate to repair the communication link. The performance index or candidate Lyapunov function is defined as:

$$V_i = \frac{1}{2}\sum_{j=1}^{N_i} k_{ip}(a_{ij} - c_{ij})^2 + \frac{1}{2}k_{iv}||v_i||^2 \tag{3.22}$$

where v_i is the velocity, c_{ij} is the desired RSSI value, k_{ip} is the potential energy and k_{iv} is the kinetic energy of the robot. The control input is derived by:

$$u_i = -\frac{\partial V_i}{\partial a_i} - \frac{\partial V_i}{\partial v_i} \tag{3.23}$$

$$u_i = -\sum_{j=1}^{N_i} k_{ip}(a_{ij} - c_{ij})\frac{p_{ij}}{a_{ij}} - k_{iv}v_i \tag{3.24}$$

According to the authors, these are the necessary conditions for controlling R_i to keep the connection. In order to evaluate their approach, the authors proposed two scenarios: Self-configuration and Self-healing; and Tethering and Intelligent Relays. Tethering is the process for which the robot network become a chain, stretching in order to enable a robot to reach the most away point, in a way that each robot has at most two neighbours. While self-healing or self-configuration the robots should pass from a disconnected to a connected configuration.

Despite proposing a distributed algorithm for tethering and self-healing, this approach rely on a central unit for managing the robots in a centralized fashion. Their work is main characterized by evaluating real metrics (such as RSSI) instead of distance, on the link quality and stability of their control scheme. Despite that, the team of robots need a leader from who the commands are given.

The authors of [40] propose the use of m UAVs to provide communication network for a group of ground vehicles. Aerial vehicles are controlled via a generalized gradient decent method, where the controller takes the form:

$$\dot{x_i} = -Ln(\partial H)(x_i) \tag{3.25}$$

where $Ln(\partial H)(x_i) : R^d \rightarrow R^d$ is the generalized gradient vector field, and $-Ln(\partial H)(x_i)$ is the direction of decent of H at $x_i \in R^d$. Their cost function incorporates a measure of signal known as the Signal-to-Interference and Noise Ratio (SINR) defined as [41]:

$$SINR_{ij} = \frac{f_{ij}}{N_i + \sum_{k \in \mathcal{N}_{i \diagdown j}} f_{ij}} \tag{3.26}$$

where f_{ij} is the communication strength (defined further) over the link $i-j$, $\mathcal{N}_{i \diagdown j}$ is

the set of neighbors of i not including j, and N_i is the environmental noise around i. They defined f_{ij} as:

$$f_{ij} = \begin{cases} \frac{P_0}{dist_{ij}^{\alpha}+1} - C & , dist_{ij} < R_{comm} \\ 0 & , dist_{ij} \geq R_{comm} \end{cases} \tag{3.27}$$

where $C = \frac{P_0}{R_{comm}^{\alpha}}$ is a constant to ensure continuity at $dist_{ij} = R_{comm}$. It can be seen that f_{ij} has a non-smooth transition to zero at $dist_{ij} = R_{comm}$, this is used to model loss of communication between two vehicles over distances greater than R_{comm}. The cost function H is then defined by the authors as:

$$H = \sum_i \sum_{j \neq i} -SINR_{ij} + \frac{\lambda}{SINR_{ij} + \delta} \tag{3.28}$$

where the term $\delta \in (0, 1]$ is included to give continuity to the function when agents i, j become disconnected (the value of $SINR_{ij}$ goes to 0). Despite the cost function being global, the control of each agent is local. From the equation 3.28, a higher weight on the second term contribute to agents equalize their SINR values amongst their neighbors. On the other hand, if the first term has a higher weight, the agents will greedily improve individual SINR links.

The authors present two approaches for connectivity maintenance. The first approach identifies the minimum cost of a disconnected network, and requires that the initial conditions of any network are below this value. The second approach is to find a critical value of λ in 3.28 such that the UAVs never move outside R from their neighbors.

The authors define the cost of a disconnected graph as:

$$H_d = 2(N-1)\frac{\lambda}{\delta} + \sum_{u \neq s} \sum_{w \neq s} -SINR_{uv} + \lambda(SINR_{uw} + \delta)^{-1} \tag{3.29}$$

$$H_{d_{min}} = 2N(N-1)\frac{\lambda}{\delta} - (N-1)(N-2)((P_0 - C) - \lambda((P_0 - C) + \delta)^{-1}) \tag{3.30}$$

Thus, they concluded that if the initial configuration has a cost $H_{initial} < H_{d_{min}}$ then the UAVs will remain connected all the time. In order to ensure that an agent i will not move outside the communication radius R from a neighbour j, the authors defined the value of λ_{crt}. This value is achieved making the product $\frac{\partial H}{\partial x_i}^T (x_i - x_j) = 0$. In this way, according to the authors, the agent i will never move further than the distance R away from j, $\forall j \in \mathcal{N}_i$.

In [60], the authors adapted the power iteration method of estimation of eigenvalues and eigenvectors in order to obtain a continuous-time decentralized

algorithm for estimation of λ_2 using only the information of one-hop neighbours. Using such algorithm, they propose a control scheme to increase the connectivity measure of a team of n agents.

$$\lambda_2 = \min_{x \perp \mathbf{1}, x \neq 0} \frac{x^T L x}{x^T x} = \min_{x \perp \mathbf{1}, x \neq 0} \frac{\sum_{(i,j) \in E} A_{ij}(x^i - x^j)^2}{x^T x} \tag{3.31}$$

where A_{ij} is the adjacency matrix defined according to equation 3.32.

$$A_{ij} = \begin{cases} e^{-||p^i - p^j||_2^2 / 2\sigma^2} & \text{if } ||p^i - p^j||_2 \leq r \\ 0 & \text{otherwise} \end{cases} \tag{3.32}$$

where r is the communication threshold,p^k is the position of agent k, and σ is chosen such that $e^{-r^2/2\sigma^2} = \epsilon$, with ϵ being a small predefined threshold.

In order to clarify about the motion control, the authors first observed that given the normalized eigenvector $\hat{v}_2$ (such that $||\hat{v}_2|| = 1$) corresponding to λ_2. Considering the values associated to A_{ij}, they concluded the control of agent k as

$$u^k = \sum_{(k,j) \in E} -A_{kj}(\tilde{v}_2^k - \tilde{v}_2^j)^2 \frac{p^k - p^j}{\sigma^2} \tag{3.33}$$

The authors conducted a simulation in a 2D environment with a group of 6 nodes, 3 playing the role of leaders and 3 followers. The leader robots used the same sinusoidal motion model given by the equation 3.34, while the followers used the control law in 3.33. According to the authors, the follower robots could move in the environment maintaining the connectivity.

$$\begin{aligned} \dot{p}_x^i(t) &= -0.2 \\ \dot{p}_y^i(t) &= 0.5cos(p_x^i) \end{aligned} \tag{3.34}$$

Lindhé *et. al.* [61] addressed the problem of coordinating a set of N robots with their path established *a priori*, such that the agents could maintain visual connectivity for all times during the path traversal in a space with obstacles [61] . They handled the problem using a rapidly exploring random trees (RRT) algorithm. In their approach, they proposed a outage detector algorithm which calculates the amount of success from a configuration q_i towards a given point y before any link of the system could be blocked by an obstacle. In their findings, they observed that because the RRT algorithm is only probabilistically complete, i.e. the method cannot in finite time conclude that a problem is unsolvable, the stop procedure decision is still heuristic.

In [62], the authors presented an extension to the Darwinian Particle Swarm

Optimization (DPSO) algorithm, which they named Robot DPSO (RDPSO), incorporating three key aspects: a) social exclusion and inclusion; b) obstacle avoidance; and c) ensuring MANET[2] connectivity. The Darwinian PSO [63] is an extended version of the traditional PSO algorithm in a way that natural selection or survival-of-the-fittest is added in order to enhance the chances for escaping from local optima. In the RDPSO, each agent in the swarm is represented by a particle. As inherited by the PSO, the particles move in a multidimensional space in a certain velocity. Each particle position and velocity are denoted by $x_i[t]$ and $v_i[t]$, respectively. Those vectors are updated using the following rules:

$$\begin{aligned} v_i[t+1] = c_0 v_i[t] &+ c_1 r_1(\hat{g}_i[t] - x_i[t]) \\ &+ c_2 r_2(\hat{x}_i[t] - x_i[t]) \\ &+ c_3 r_3(\hat{X}_i[t] - x_i[t]) \\ &+ c_4 r_4(\hat{X}_i^m[t] - x_i[t]) \end{aligned} \tag{3.35}$$

$$x_i[t+1] = x_i[t] + v_i[t+1] \tag{3.36}$$

where r_i is a random vector with entries in the interval $[0, 1]$, c_0, c_1, c_2, c_3 and c_4 are weights for the inertial influence, the global best, the local best and the obstacle avoidance, communication ensuring , respectively. $\hat{X}_i^m$ is the position of the nearest neighbour of robot $i + R_{comm}$ (the communication radius).

In order to handle the connectivity, the authors utilize the multi-hop paradigm. In such model, they use the sum of powers of the adjacency matrix to capture the broken links. Despite the multi-swarm approach, the authors highlight that every robot need to be aware of the signal quality or position of all other swarm members. This is mandatory to calculate the matrix of connectivity. Their approach was validated experimentally through simulations with robots being deployed in an outdoor scenario of 300x300 meters. In this scenario, the obstacles were deployed randomly with a non-defined regular density. Since the RDPSO is stochastic, they took the data from 100 trials with 300 iterations. The number of swarms was varied from 1, 3 and 6. The search objective is represented by a Gaussian distribution.

Furthermore, beyond the no-limited (i.e. $R_{comm} \to \infty$) scenario, three other connectivity technologies were evaluated: WiFI, ZigBee and Bluetooth, which in practice represents an assignment to the R_{comm} value respectively of 100m, 55m and 10m. Under the exposed scenario, their approach showed that for the no-limited situation, it was necessary at least 15 robots to get to the maximum of the objective search, followed by up to 21 robots using WiFi, and up to 27 robots to the ZigBee

[2]MANET stands for Mobile ad hoc network.

and Bluetooh cases. It is important to report that the max network neighboring capabilities of such network technologies have not been considered. For example, each network (piconet) using the Bluetooth can only be formed by 8 nodes.

In [64], the authors handled the formation problem reconfiguration of multi-UAVs under more realistic physical dynamics of the UAVs. (formation reconfiguration problem definition here). They formulated the reconfiguration problem as a parameterized optimal control problem. For the parameterization, they used the Control Parameterization and Time Discretization (CPTD) method proposed by [65]. The authors modelled the formation control using the energy function exposed in the following equation:

$$\begin{aligned} J = \min_{\Omega, \Delta t_p} \{ (n_p \Delta t_p) + \sigma g_1(\Omega, \Delta t) \\ + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} [\sigma_{ij} \max(0, R_{safe} - d_{ij}(x_i(t), x_j(t))) \\ + \sigma'_{ij} \max(0, d_{ij}(x_i(t), x_j(t)) - R_{comm})] \end{aligned} \tag{3.37}$$

where Ω and Δt_p are the parameterized control matrix and the time interval in the use of the piece-linear function approximation of the non-linear control. The constraint g_1 limits each entry on the control matrix Ω to respect the limits defined by the control inputs. R_{safe} is the safe distance for which the UAVs should maintain to avoid impact. Furthermore, R_{comm} is the communication radius and the σ parameters are appropriate weights.

In order to solve the reconfiguration formation problem, Duan *et.al* [64] developed a hybrid PSO and Genetic Algorithm to minimize the energy function defined in (3.37). The authors validated their approach experimentally using a 6-degree UAV dynamic function. It is important to highlight that there is no topology changing protocol in their approach. By the equation (3.37), it can be seen that the only considered topology is the complete graph.

3.3 Final remarks about the connectivity problem

Based on the aforementioned works, it can be concluded that the positioning of a group of agents in order to keep the global connectivity is still an active research field. Therefore, it is believed that there are still a lot of work towards the development of new methods and control schemes for the connectivity problem. In this thesis, it is considered a discrete time model where some computational approaches are proposed in order to deal with the connectivity maintenance in a tracking system where a group of pursuers want to capture/follow a group of targets, and the connectivity

is maintained through the positioning of a set of relays. The first approach for this problem is covered in the next chapter.

Chapter 4

Quadratic Programming with linear restrictions

We consider the problem of positioning *pursuers* and *relays* in a way that minimizes the average distance between targets and pursuers, subject to movement and connectivity constraints. Our first approach consists in formulating at each stage of time a linear quadratic problem. This problem is solved by a recurrent neural network associated to its primal-dual optimality conditions.

4.1 Introduction to the model

Let $P = \{p^1, p^2, \ldots, p^N\}$ be a set of pursuers which have the mission of move towards a set of targets $\Theta = \{\theta^1, \theta^2, \ldots, \theta^N\}$, for which members have unknown trajectories with bounded velocities. Let $R = \{r^1, r^2, \ldots, r^M\}$ be a set of relays, whose mission is to position themselves in order to provide connectivity among the elements of P. We consider the time is discrete, i.e., from instant k to $k+1$ the targets make a move while the pursuers and relays have to decide over their positioning. This problem can be modeled as a quadratic programming model for which it is desired to minimize the average distance among the elements of P and Θ, while the proximity graph induced by the positioning of elements in $P \cup R$ is connected.

In order to introduce the time index, let $P_k = \{p_k^1, p_k^2, \ldots, p_k^N\}$, be the pursuers position in instant time k, such that $p_k^i \in \mathbb{R}^2$ (Cartesian Plane). Let the sets Θ_k and R_k be similarly defined. In this way, the quadratic model of the aforementioned problem is given by equations (4.1), (4.2) and (4.3).

$$\min_{i\in\{1,2,\ldots,N\}} \frac{1}{N}\sum_{i}^{N}(p_{k+1}^{i}-\theta_{k}^{i})^{2} \tag{4.1}$$

s.t:

$$(x_k - x_{k+1})^2 \leq R_{step}^2 \quad \forall x \in P \cup R \tag{4.2}$$

$$G(R_{k+1} \cup P_{k+1}) \text{ is connected} \tag{4.3}$$

where, $G(\mathbf{X})$ is the proximity graph induced by the 2D positions of the elements of $\mathbf{X}$, and R_{step} is the maximum step size an agent can travel from instant k to $k+1$.

The constraint (4.2) is straightforward for representing the movement restriction of agents, from time step k to $k+1$. On the other hand, constraint (4.3) could be cumbersome to handle. Furthermore, the constraint (4.3) does not impose restrictions on the connected classes the graph should be in. But, if we consider only the class of trees, since a tree is the connected graph with the minimum number of edges, then according to Cayley [66], the number of possible labeled trees with n vertices is n^{n-2}.

In this way, we consider a fixed tree T whose vertices are given by $1,\ldots,N+M$ which are related with the set of vehicles $P_{k+1} \cup R_{k+1}$. Therefore, the restriction (4.3) becomes:

$$\begin{aligned} &\forall (i,j) \in E(T) \\ &\quad (x_{k+1}^{i} - x_{k+1}^{j})^2 \leq R_{comm}^2 \qquad\qquad x_l \in R_{k+1} \cup P_{k+1} \end{aligned} \tag{4.4}$$

where, $E(T)$ is the set of edges of the fixed structure T and R_{comm} is the maximum communication radius.

In this first version of the proposed solution, we assume that the network structure is fixed (T) for all simulated seconds. We concentrate on solving efficiently the positioning problem. When considering the problem using as presented by the previous model (equations 4.1, 4.2 and 4.4), it leads to a class of non-linear programming called quadratic problem with quadratic constraints (QPQC). In such discipline, a good feature is to know if the problem is concave or convex in order to apply the proper algorithms to solve it. In order to discuss that, suppose the following generic QPQC problem:

$$\min_{x} x^T M_0 x + q_0^T x \tag{4.5}$$

$$s.t.:$$

$$x^T M_i x + q_i^T x \leq b_i \tag{4.6}$$

if all $M_i (i = 0, \ldots, n)$ matrices are semi-definite positive, then the problem is convex [67]. However, in this chapter we deal with a relaxed version of the problem. Instead of considering the Euclidean distance, we use the Manhattan distance. This leads to a quadratic programming model with quadratic objective and linear restrictions. This model is presented in the next section.

4.2 Proposed mathematical programming model

In the considered scenario, we consider the case in which a set of N agents (*pursuers*) are pursuing a set of N targets, on a limited Cartesian plane such as:

$$X_{min} \leq x \leq X_{max} \tag{4.7}$$

$$Y_{min} \leq y \leq Y_{max} \tag{4.8}$$

where $X_{min}, X_{max}, Y_{min}$ and Y_{max} defines the boundaries of the Cartesian plane, as well as x and y means the coordinates of a vehicle. Furthermore, we also consider positioning a set of M mobile connecting devices (*relays*) which are used to maintain communication between the *pursuers*. It is supposed that to each pursuer p_i is assigned a target $t_i, i = 1, \ldots, N$, while the communication structure is given by a tree (T) whose nodes are relays $r_j, j = 1, \ldots, M$ and the leaves of T are pursuers as shown in Figure 4.1.

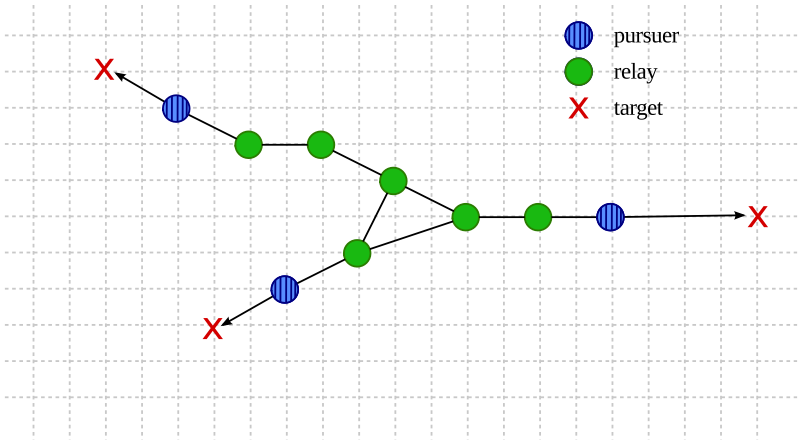


Figure 4.1: Considered scenario.

Here, the time is discretized and at time $k-1$ the positions of the targets are given by their Cartesian coordinates $\theta^i_{k-1} = (X^i_{k-1}, Y^i_{k-1}), i = 1, \ldots, N$ while the positions of the pursuers are given by their Cartesian coordinates $p^i_{k-1} = (x^i_{k-1}, y^i_{k-1}), i = 1, \ldots, N$ as well as the position of the relay nodes $r^i_{k-1} = (x^i_{k-1}, y^i_{k-1}), i = N + 1, \ldots, N + M$.

Consider that the connecting tree T composed of $N + M$ vertices. Therefore, it

is considered that connectivity is achieved at time k when:

$$\text{for each } (i,j) \in E(T): \tag{4.9}$$

$$-R_{comm} \leq x_k^i - x_k^j \leq R_{comm} \tag{4.10}$$

$$-R_{comm} \leq y_k^i - y_k^j \leq R_{comm} \tag{4.11}$$

whereR_{comm} is the maximum communication threshold and $E(T)$ is the set of edges of the tree T, since this structure does not change over time, it is not indexed by the k variable.

It is also supposed that from a time period to the next, the position change of the targets is limited in a way such as for $i = 1, \ldots, N$:

$$-\delta \leq X_{k-1}^i - X_k^i \leq \delta \tag{4.12}$$

$$-\delta \leq Y_{k-1}^i - Y_k^i \leq \delta \tag{4.13}$$

where $\delta < R_{comm}$ is the maximum step size of target i while

$$X_l^i \in [X_{min}, X_{max}], Y_l^i \in [Y_{min}, Y_{max}], for \quad i = 1, \ldots, N \tag{4.14}$$

Now we consider that the pursuers take positions at stage k such as they minimize the performance index given by:

$$\bar{d_k} = \frac{1}{N} \sqrt{\sum_{i=1}^{N} (x_k^i - X_k^i)^2 + (y_k^i - Y_k^i)^2} \tag{4.15}$$

which is the mean distance between pursuers and targets, while satisfying the above constraints (4.9) to (4.14) and a set of step size limitations constraints expressed as:

$$-R_{step} + x_{k-1}^i \leq x_k^i \leq R_{step} + x_{k-1}^i \tag{4.16}$$

$$-R_{step} + y_{k-1}^i \leq y_k^i \leq R_{step} + y_{k-1}^i \tag{4.17}$$

where R_{step} is the step size limitation of the agents from one instant of time to another.

Then the solution of this problem at stage k is identical to the solution of the

linear quadratic mathematical programming problem given by:

$$min_{x_k^i, y_k^i, i=1,\ldots,N} \sum_{i=1}^{i=N} (x_k^i - X_k^i)^2 + (y_k^i - Y_k^i)^2 \tag{4.18}$$

under constraints (4.9), (4.12), (4.13), (4.14),(4.16), and (4.17).

4.3 Recurrent neural network as a solver

The basic idea for solving an optimization problem using a tailored neural network is to make sure that the neural network will converge asymptotically at a fast rate and that the equilibrium point of the neural network will correspond effectively to the solution of the original optimization problem. In 1986, Tank and Hopfield introduced a linear programming neural network solver realized with an analogic circuit which appeared to be well suited for applications requiring on-line solutions [68]. After this first successful attempt, many neural network models for solving linear and quadratic programming problems have been proposed in the literature. For a review see [69, 70].

According to the relationship between the states of the neural network and the values of primal and dual decision variables, it is possible to divide the existing recurrent neural network for solving linear and quadratic programming problems into three classes: primal neural network, primal-dual neural network, and dual neural network.

In the present case, the mathematical programming problem presents inequality constraints as well as bounding limits. The adoption of a primal-dual neural network leads to add various slack variables, turning the size of network larger. This primal-dual neural network is built such as global convergence is guaranteed while the convergence speed can be adjusted by choosing an adequate value for its learning parameter [7]. To display the structure of the linear quadratic neural network solver, a general linear-quadratic programming problem is parameterized as follows:

$$\min f(\delta) = \frac{1}{2}\delta^T Q\delta + c^T\delta \tag{4.19}$$

$$\text{s.t. } h(\delta) = J\delta - d = 0 \tag{4.20}$$

$$g(\delta) = A\delta - b = 0 \tag{4.21}$$

$$\xi^- \leq \delta \leq \xi+ \tag{4.22}$$

δ is the decision vector, representing in our case the positions of the swarm entities. Matrix Q is assumed symmetric positive semi-definite which allows to handle in a

similar way linear quadratic and linear programming problems.

Once constraints (4.20), (4.21) and (4.22) are feasible, at least one optimal solution $\delta*$ will meet the Karush-Kuhn-Tucker optimality conditions (KKT) [71]. Then equations (4.20) to (4.22) can be turned equivalent to the following set of linear variational inequalities:

$$(y - y*)^T(Hy* + p) \geq \mathbf{0} \forall y \in \Omega \tag{4.23}$$

with the primal-dual variables $y = [\delta^T \quad u^T \quad v^T]^T$. Then the problem is to find a solution vector $y*$. Its feasible region Ω and its lower/ upper limits are given by:

$$\Omega := \{y|\zeta^- \leq y \leq \zeta+\} \quad \zeta^- = [\xi^- \quad -\omega^+ \quad 0]^T \quad \text{and } \zeta^+ = [\xi^+ \quad \omega^+ \quad \omega^+] \tag{4.24}$$

Here ω^+ has an appropriate dimension and each of its entries is chosen to be sufficiently large to replace $+\inf$ numerically. The coefficients are defined as:

$$\rho = [c^T \quad -d^T \quad b^T]^T \text{ and } H = \begin{bmatrix} Q & -J^T & A^T \\ J & 0 & 0 \\ -A & 0 & 0 \end{bmatrix} \tag{4.25}$$

Then the neural network model which solves (4.19) with (4.20), (4.21) and (4.22) is given by :

$$\frac{dy}{dt} = \lambda(E + H^T)\{P_\Omega(y - (Hy + \rho)) - y\} \tag{4.26}$$

where λ is a positive learning parameter which can be used to adjust the convergence speed of the network, E is an identity matrix, $P_\Omega[\cdot]$ is a piecewise-linear function defined in (4.27). Figure 4.2 shows the block diagram of the implementation of this neural network.

$$P_\Omega[y_i] = \begin{cases} \zeta_i^-, & \text{if } x_i \leq \zeta_i^- \\ \zeta_i^+ & \text{if } x_i \geq \zeta_i^+ \\ y_i, & \text{otherwise} \end{cases} \tag{4.27}$$

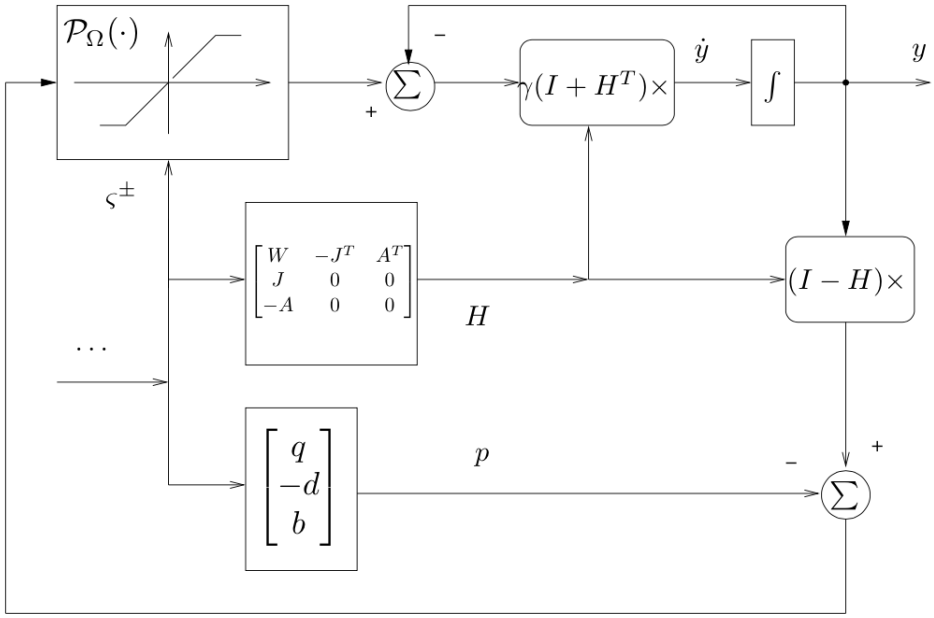


Figure 4.2: A diagram of the recurrent neural network solver proposed by Zhang et.al [7].

4.4 Experimental Simulations

As to see the velocity of the neural network as a solver to the current quadratic programming problem with linear equations, it has been run an experiment comparing the quadratic solver of Matlab and the performance of the neural network under some test scenarios with up to 500 robots (which implies at least in double of variables in the model). The recurrent neural network has been simulated using the Simulink software which comes embedded into Matlab. In all tests, the pursuers and relays are randomly deployed with coordinates in the range of -1.0 to 1.0. Over those, a minimum spanning tree is created using edges based on the Euclidean distance and this network structure is fixed during all the simulation.

The first test set consisted of using 10 pursuers and 4 relays. In such scenario there was 10 targets which were deployed in random points. The moving strategy of such targets is random walks over the space. This scenario has been simulated over 5 runs and the time information is given by Table 4.1. Figures 4.3 and 4.4 show a convergence graph of one of the runs using the aforementioned scenario and a screenshot of the simulation, respectively.

Simulation	Quadprog	RNN convergence time
1	0.29s	$< 0.03s$
2	0.36s	$< 0.03s$
3	0.24s	$< 0.03s$
4	0.31s	$< 0.03s$
5	0.29s	$< 0.03s$
Average Time	0.29s	$< 0.03s$

Table 4.1: Scenario with 10 pursuers, 10 targets and 4 *relays*, behaviour: *random walks* e 10 movement steps.

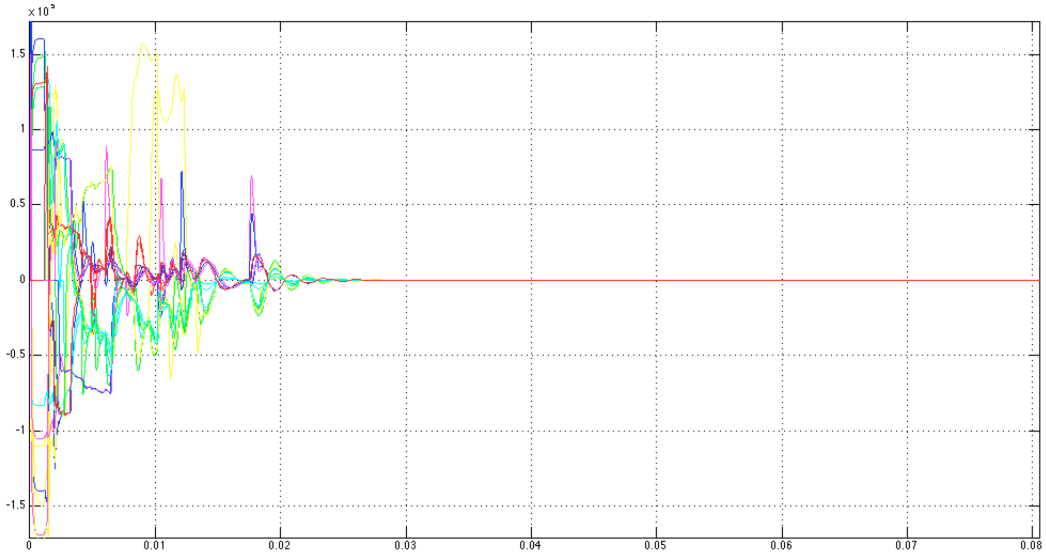


Figure 4.3: Convergence time of the recurrent neural network over the data on Table 4.1

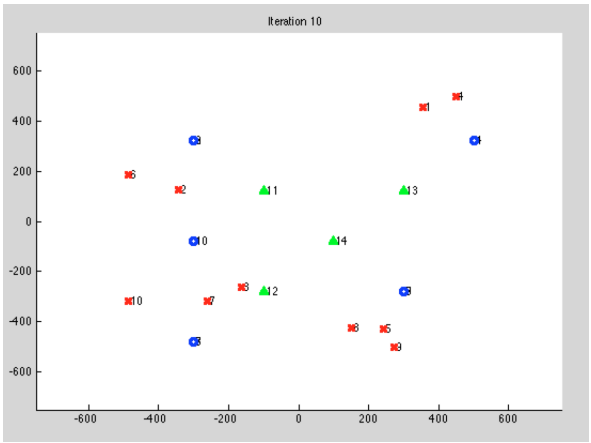


Figure 4.4: Screenshot of the experiment with 10 pursuers.

In the next scenario, there has been a slight increase on the number of robots. An amount of 35 pursuers and targets as well as 10 relays, using the random walks

for the targets during 25 seconds of simulation. The results are shown on Table 4.2. A graph of the neural network convergence of one of the simulations is exposed on Figure 4.5.

	Quadprog	**RNN convergence rate**
1	12.97s	$< 0.15s$
2	17.40s	$< 0.15s$
3	12.55s	$< 0.15s$
4	13.14s	$< 0.15s$
5	11.04s	$< 0.15s$
Average time	13.42s	$< 0.15s$

Table 4.2: Scenario with 35 pursuers/targets and 10 relays over 25 seconds of simulation.

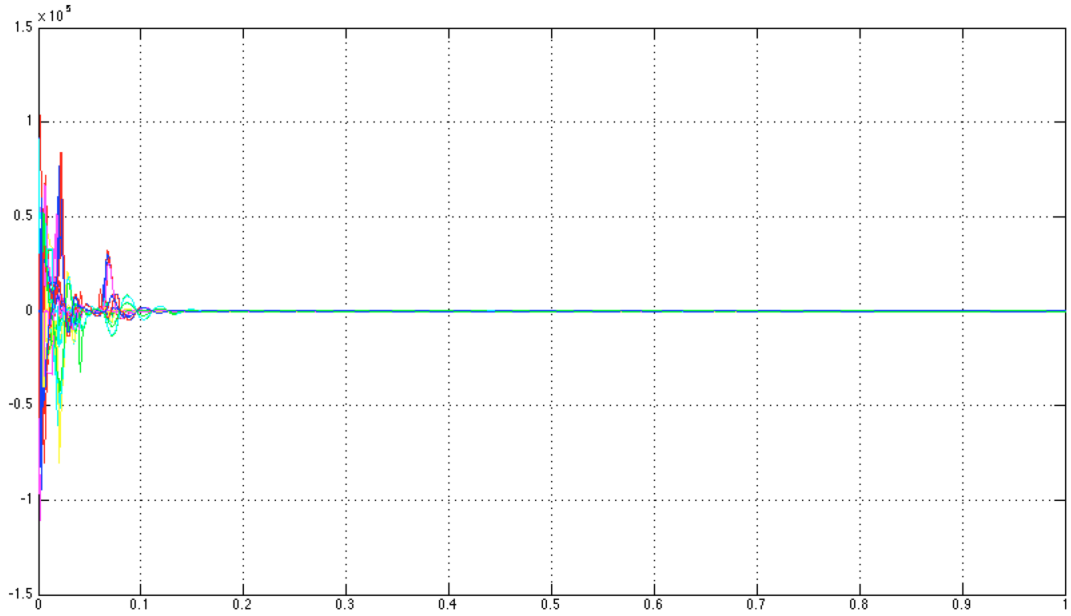


Figure 4.5: RNN convergence time of one of the simulations shown on Table 4.2

To finish the numerical simulations, one last scenario had been tried: a setup using 400 pursuers, 400 targets and 100 relays, with targets moving using random walks. Over this setup, an amount of 50 seconds had been simulated. The results are shown on Table 4.3.

	Quadprog	RNN convergence time
1	57610s(≈16h)	$< 0.35s$
2	61595s(≈17h23m)	$< 0.35s$
3	55288s(≈15h21m)	$< 0.35s$
Average time	58164s(≈16h10m)	< 0.35

Table 4.3: Scenario with 400 pursuers/targets and 100 relays during 50 seconds of simulation.

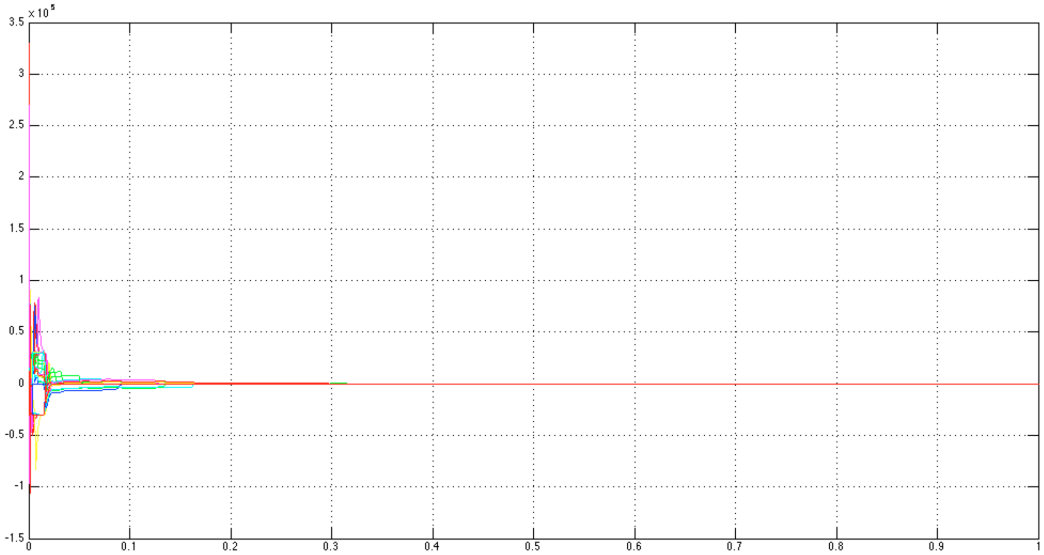


Figure 4.6: Convergence time of one second of simulation over the scenario present in Table 4.3

Observing the data exposed on Table 4.3, it can be seen that in the average, the RNN has converged to the correct values of the considered quadratic model in less than $0.35s$ for each simulated second. Considering the total amount of simulated seconds and considering only the convergence time, this gives a total of $17,5s$ against $58164s$ from the `quadprog` solver.

4.5 Conclusion

By the experiments exposed in the previous section, it can be seen that the neural network solver time of simulation was faster than the `quadprog` procedure, and even faster when considering a scenario with a considerable amount of variables. Despite the neural network solver convergence time is so reduced, this only solves the problem of positioning the pursuers and relays over a static structure. Furthermore, there is an error in the correct positioning because it has been used the Manhattan distance instead of the Euclidean one. This was necessary in order to deal with a quadratic

problem with linear constraints and therefore be dealt using QP solvers and the recurrent neural network used for comparison. In the next chapter, we present a surrogate approach for dealing with the dynamic structure of the network using the Euclidean distance as well.

Chapter 5

Collective intelligence of a swarm of pursuers

In this chapter, we present our approach for dealing with the pursuing mission using a dynamic network setting. We begin by shortly defining the problem in Section 5.1 and then depicting the solution strategy in Section 5.2. In such section we define the estimation method of λ_2 (the algebraic connectivity) and furthermore we define a distributed solution to finding the new positioning of both *pursuer* and *relay*. This distributed approach is based solely on the network neighborhood of the node. At first, we thought that the minimum value of the proposed measure was the one between the minimum graph (two nodes and one edge). We found out that the minimum value of the proposed connectivity index follows a certain pattern. This subject is discussed in Section 5.2.2. The ordering to parallel the computation of the computation of each node is discussed in 5.2.3. We finish the chapter by writing about the complexity of the positioning computation (Section 5.3).

5.1 Considered scenario

Let N_p vehicles evolving on the plane whose mission is to reach a set of N_t moving targets with the assistance of N_r mobile relays which maintain connectivity between them so that they can coordinate their behavior. Time is discretized: every period decisions are taken for relays and pursuers. It is supposed that the position change of a mobile during the period is bounded. Connectivity assurance is a centralized function while pursuing targets are decentralized functions. Here connectivity makes feasible collective intelligence for the swarm of relays and pursuers.

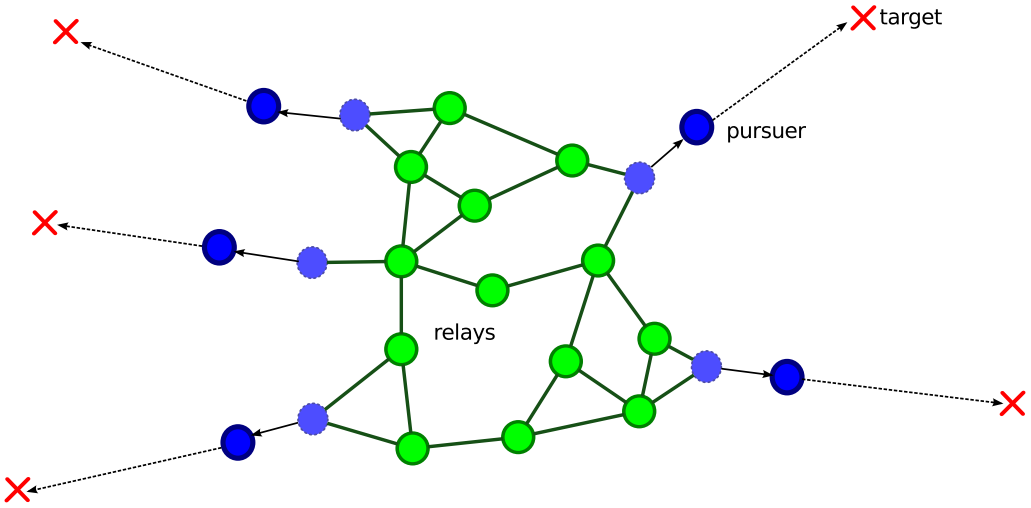


Figure 5.1: Initial environment of the considered problem.

5.2 Solution strategy

Let $N_{PA} = |N_{PA}|$ be the number of *active pursuers* to which are assigned targets to be pursued. At start this number is taken equal to $\min\{N_p, N_t\}$. At time step $k = k_0 + k.\Delta t$ the N_{PA} pursuers provide the estimated positions of their targets at time $k+1$: $\hat{\theta}^i_{k+1}$. Note that this can be done by each pursuer considering the speed of its target during the last periods, its limitations (minimum turn radius, etc and local obstacles).

Let p^i_k be the position of the i^{th} pursuer at time k, let be $\hat{\theta}^i_k$ the estimated position of target i at time k, and R_{step} being the maximum step size the vehicle can reach from k to $k+1$, The central decision maker will solve the following problem:

$$\text{Maximize } \alpha \tag{5.1}$$

$$\text{subject to:}$$

$$\hat{p}^i_{k+1} = p^i_k + \alpha.(\hat{\theta}^i_{k+1} - p^i_k) \tag{5.2}$$

$$dist(\hat{p}^i_{k+1} - p^i_k) \leq R_{step} \tag{5.3}$$

$$\alpha > 0 \tag{5.4}$$

while the connectivity is maintained. This can be considered to be a bilevel optimization problem since the connectivity condition can be traduced by the solution of another optimization problem. Let S_k be the solution associated to α^*. If $\alpha^* > 0$, then let $\hat{r}^i_{k+1}$ be the position of the closest relay to the i^{th} pursuer in S_k and let $\rho^i_k = ||\hat{p}^i_{k+1} - \hat{r}^i_{k+1}||$ be the feasible radius.

The pursuers will choose their position in a decentralized way at time $k+1$ by: $p^i_{k+1} = \hat{r}^i_{k+1} + \rho^i_k.(\hat{\theta}^i_{k+1} - \hat{r}^i_{k+1})/||(\hat{\theta}^i_{k+1} - \hat{r}^i_{k+1})||$, except if $||(\hat{\theta}^i_{k+1} - \hat{r}^i_{k+1})|| < \rho^i_k$ when the i^{th} target is considered to be reached by the i^{th} pursuer. Let $\Delta p^i_k(\alpha) = p^i_{k+1} - p^i_k, \forall i \in$

N_{PA}. If $\alpha^* = 0$, it means that no global progress towards the targets is feasible. A strategy in that case can be to cancel temporarily the pursuit of some targets and to concentrate the pursuit on a reduced number of targets. Here it is proposed to delete from the list of pursued targets the farthest target from its pursuer. Then this pursuer might play momentarily the role of a relay or other available role in order to enhance the support to the pursuit of the remaining active targets or even to play the role of a passive pursuer, for which the movements would only direct to its assigned relay. Once a target is reached, the closest passive target to a potential pursuer (those pursuers which are not assigned to a target) will become active again and the closest pursuer will be assigned to it.

This process will continue until all targets have been reached or other criteria (such as the maximum number of elapsed simulation seconds is run out) may be reached as well. In the case in which the targets have different degrees of criticality, the choice of the target to delete/include in the active set of pursued target can be based on these degrees.

5.2.1 Connectivity

Instead of considering the positivity of λ_2, the first positive eigenvalue of the Laplacian matrix attached to the network, as a connectivity algebraic criterion, observe that if λ_2 is not null, 0 is not a double root of the characteristic polynom (Figure 5.2) defined here as $\pi(s) = \det(L - sI)$ attached to the Laplacian matrix of the proximity graph induced by the positions of the vehicles. Then an alternative algebraic criterion for connectivity is the value of $\pi'(s)$ at $s = 0$, and to get a positive index, consider $c^2 = (\pi'(s))^2$. Observe that c can be computed directly from L without deriving $\pi(s)$:

$$\pi'(0) = \sum_{i=1}^{N} \det L(i) \tag{5.5}$$

where $L(i)$ is equal to L except the i^{th} column which is equal to zero with a -1 in the i^{th} row. Another surrogate approach is to consider a second order approximation of the characteristic polynomial of the Laplacian matrix at the origin:

$$\pi(s) = \pi(o) + \pi'(0).s + \frac{1}{2}\pi''(0).s^2 + O^3(s) \tag{5.6}$$

with $\pi(o) = 0$ and an estimate of λ_2 is given by:

$$l_2 = 2.\pi'(0)/\pi''(0) \tag{5.7}$$

where:

$$\pi''(0) = 2.\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \det L(i,j) \tag{5.8}$$

with $L(i,j)$ is equal to L except the i^{th} column which is equal to zero with a -1 in the i^{th} row and idem with column j.

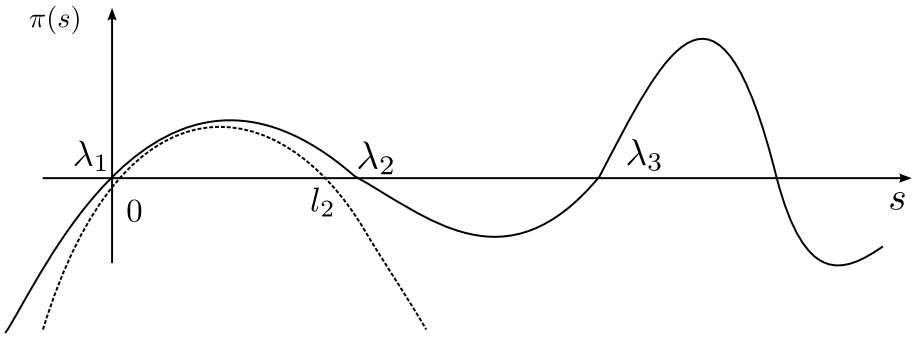


Figure 5.2: Estimation of λ_2.

Consider that $N = N_p + N_r$ with $N_r \geq N_p$. It is supposed that the first N_p relays are connected with a unique pursuer which is not connected to other relays. Then the structure of L is as follows:

$$L = \begin{bmatrix} W_{P\times P} & -W_{P\times P} & 0_{P\times(R-P)} \\ -W_{P\times P} & L_{R\times R} - \begin{bmatrix} W_{P\times P} & O_{P\times(R-P)} \\ 0_{P\times(R-P)} & 0_{(R-P)\times(R-P)} \end{bmatrix} & \\ 0_{(R-P)\times P} & & \end{bmatrix} \tag{5.9}$$

where $W_{P\times P}$ is diagonal and $L_{R\times R}$ is the Laplacian matrix associated to the connectivity of the relays.

Given the new positions of the pursuers, $p_k^i + \Delta p_{k+1}^i(\alpha)$, $i = 1 \ldots N_p$, the connectivity problem can be formulated as find $\{\delta r_k^i | i = 1, \ldots N_r\}$ such that: $c^2(r_k + \delta r_{k+1}^i) > c_{min}^2$ with $||p_k^i + \Delta p_{k+1}^i(\alpha) - (r_k^i + \delta r_k^i)|| \leq R_{max} \qquad i = 1, \ldots, N_p$ and $||\delta r_{k+1}^i|| \leq R_{max}, i = 1, \ldots, N_r$. Defining $\delta r_k^i(\mu) = \mu.\frac{\partial c^2}{\partial r_k^i}|_{r_k}, i = 1, \ldots, N_r$ with $\mu > 0$, search $\max \mu$ such that:

$$\max \mu \tag{5.10}$$

$$s.t:$$

$$c^2(g^{'}) \geq c_{min}^2 \tag{5.11}$$

$$||p_k^i + \Delta p_k^i(\alpha) - (r_k^i + \delta r_k^i(\mu))|| \leq R_{max} \quad \text{if } i \in N_{PA} \tag{5.12}$$

$$||\delta r_k^i(\mu)|| \leq R_{max} \tag{5.13}$$

where g' is the proximity graph induced by the new positioning of the vertices, c_{min}^2 is the the minimum value of c^2 for which the graph is still connected.

Since the computation of the derivatives $\frac{\partial c^2}{\partial r_k^j}|_{r_k} \qquad j = 1, \ldots, N_r$, over the whole network appears cumbersome, a subnetwork can be defined around each vertex j as well as the associated local Laplacian L_j and the corresponding characteristic polynomial $\pi_j(0)$ for which is attached to $\pi_j^{'}(0)$ given by:

$$\pi_j^{'}(0) = \sum_{i=1}^{N} \det\left(L_j(i)\right) \tag{5.14}$$

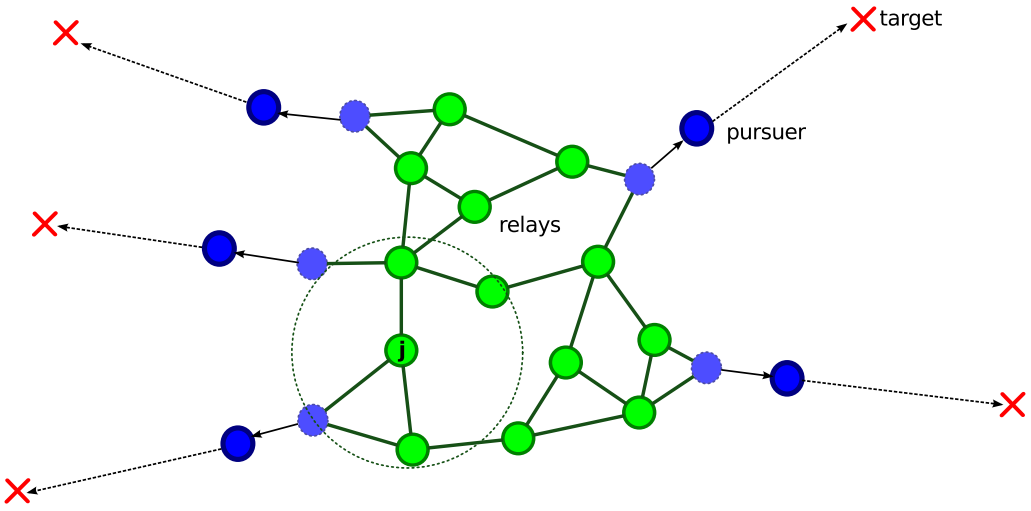


Figure 5.3: The selected subnetwork around the vertex j.

Observe that a necessary condition for global connectivity of a network is the local connectivity of each subnetwork attached to each vertices. Let $\mathcal{N}_j = \{i | dist(x_i - x_j) \leq \rho.D_{max}\}$ be the neighborhood of j, with $\rho \geq 1$, $dist(\circ)$ a distance function and x_l the state of vehicle l. The sufficient condition for local connectivity is given here by:

$$c_j^2 = (\pi'_j(0))^2 \geq c_{min}^2(\mathcal{N}_j \cup j) \qquad j \in N_{PA} \cup N_r \tag{5.15}$$

where $c_{min}^2(\mathcal{N}_j \cup j)$ is the minimum value attached to the proximity graph induced by the position of $\mathcal{N}_j \cup j$.

Then an estimation of $\frac{\partial c^2}{\partial r_k^j}|_{r_k} \quad \forall j = 1, \ldots, N_r$ can be computed more easily from this subnetwork. This leads to propose a decentralized scheme to search for new positions for the relays. Defining $\delta r_k^j = \mu_j . \frac{\partial c^2}{\partial r_k^j}|_{r_k}$ for $j \in N_{PA} \cup N_R$, with $\mu_j \geq 0$, we

model the problem as:

$$\max \mu_j \tag{5.16}$$

$$s.t: \tag{}$$

$$c_j^2(g_j^{'}) \geq c_{min}^2(\mathcal{N}_j \cup \{j\}) \tag{5.17}$$

$$||p_k^j + \Delta p_k^j(\alpha) - (r_k^j + \delta r_k^j(\mu_j))|| \leq R_{max} \quad \text{if } j \in N_{PA} \tag{5.18}$$

$$||\delta r_k^j(\mu_j)|| \leq R_{max} \tag{5.19}$$

where $g_j^{'}$ is the proximity graph obtained by the positions of $\mathcal{N}_j$ and the new position of j changed according to $r_k^j + \delta r_k^j(\mu_j)$, $c_{min}^2(\mathcal{N}_j \cup j)$ is the minimum value of c^2 measure on the structure formed by $\{j\} \cup \mathcal{N}_j$, considering the minimum value of the weighting function as the weight of each edge.

In order to move the position of node j (represented by r^j) from time k to $k+1$, we adopt $r_{k+1}^j = r_k^j + \delta r_k^j(\mu_j)$ for $j \in N_{PA} \cup N_R$. We then apply an ordering on the computation of each $j \in N_{PA} \cup N_R$, as the position of the relay providers ($j < N_{PA}$) is firstly calculated. This procedure is what we call *SwarmLambda2*.

5.2.2 A discussion about the minimum of C^2

When considering the c_{min}^2, appearing in constraint (5.17), we thought c_{min}^2 could be defined by the value attached to connectivity index for a subgraph composed of only two vertices separated by a distance of R_{max}. Let wf be a decreasing weighting function, and L be the Laplacian matrix associated to a graph with only two vertices positioned at a distance of R_{max} and containing only one edge between them. Therefore the minimum $c^2(L)$ would be:

$$L = \begin{bmatrix} a & -a \\ -a & a \end{bmatrix} \tag{5.20}$$

with $a = wf(R_{max})$ (weighting function minimum value). It turns out that the c_{min}^2 can be less than the previous value. As it will be further described, for each subnetwork structure there is a minimum value of c^2 attached to it.

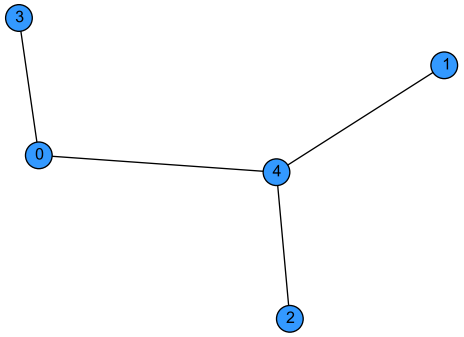
It is true that the number of labeled trees with n vertices is n^{n-2} [72]. Since the number of labelled trees is straightforward by Caley's formula, according to [66] the number of unlabelled trees (T_n) with n vertices is even more difficult to handle. Since each labelled tree can be labelled in at most $n!$ ways and the number of labelled trees is n^{n-2}, in the same reference, the authors state that the number of unlabelled trees is at least $n^{n-2}/n!$. Of course this is only a lower bound on the number of unlabelled trees. In the same reference, the authors find that an upper bound for

T_n is 4^{n-1} based on the number of of possible planar codes. Then, it is stated that

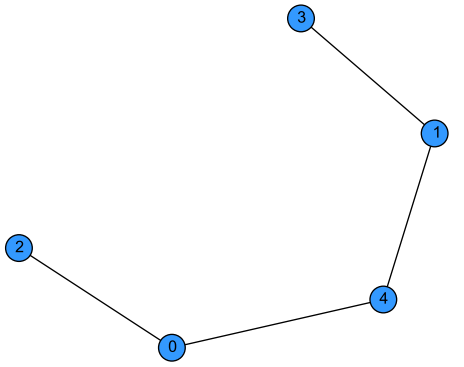
$$\frac{n^{n-2}}{n!} \leq T_n \leq 4^{n-1}$$

Suppose a is the minimum value given by the weighting function at the limit of the connection. In graph theory, it is known that a connected graph with n vertices that contain the minimum number of edges is a tree. Therefore, we first started searching for a database of all unlabelled trees of n vertices. In such search, it has been found a database with all unlabelled trees up to 22 vertices at `https://cs.anu.edu.au/people/Brendan.McKay/data/trees.html`[1]. Figure 5.4 shows all unlabelled trees (the labels on vertices is only to indicate the columns position of the corresponding Laplacian matrix) with $n = 5$ nodes. In Table 5.1, it is shown the results of the c^2 measure for trees with $n = 4 \ldots 10$ vertices. In Figure 5.4 it is shown the unlabelled trees with 5 vertices followed by the corresponding Laplacian matrix with minimum values for connectivity.

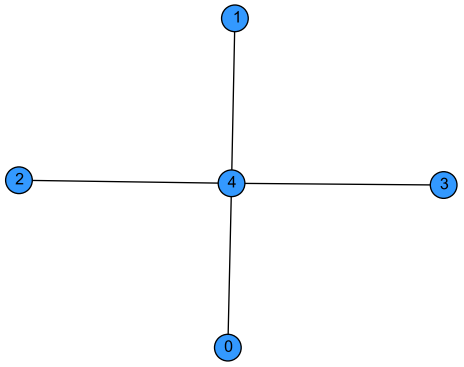
[1]The author would like to thank professor Claudia Justel at IME-RJ for such indication.



$$\begin{bmatrix} 2a & 0.0 & 0.0 & -a & -a \\ 0.0 & a & 0.0 & 0.0 & -a \\ 0.0 & 0.0 & a & 0.0 & -a \\ -a & 0.0 & 0.0 & a & 0.0 \\ -a & -a & -a & 0.0 & 3a \end{bmatrix}$$



$$\begin{bmatrix} 2a & 0.0 & -a & 0.0 & -a \\ 0.0 & 2a & 0.0 & -a & -a \\ -a & 0.0 & a & 0.0 & 0.0 \\ 0.0 & -a & 0.0 & a & 0.0 \\ -a & -a & 0.0 & 0.0 & 2a \end{bmatrix}$$



$$\begin{bmatrix} a & 0.0 & 0.0 & 0.0 & -a \\ 0.0 & a & 0.0 & 0.0 & -a \\ 0.0 & 0.0 & a & 0.0 & -a \\ 0.0 & 0.0 & 0.0 & a & -a \\ -a & -a & -a & -a & 4a \end{bmatrix}$$

Figure 5.4: In the left, it is shown all the unlabelled trees with 5 vertices. In the right, it is shown the corresponding Laplacian matrix with the minimum connectivity value for each edge represented by a (the trees are labelled only to identify the Laplacian matrix).

Table 5.1: Showing the c^2 measure over the families of unlabelled trees

# vertices	$\mathbf{c^2}$
4	$16.0a^6$
5	$25.0a^8$
6	$36.0a^{10}$
7	$49.0a^{12}$
8	$64.0a^{14}$
9	$81.0a^{16}$
10	$100.0a^{18}$

From the experiment, it is easy to see that the c^2 follows a certain pattern, which is $n^2a^{2(n-1)}$, for all unlabelled trees evaluated in the aforementioned interval. Despite the findings of the experiments, further investigation over this measure and its minimum, as well as mathematical proofs are subjects for future research.

5.2.3 Parallelism issues on the calculation of c^2

When a node j is chosen to do the computation of the direction to be taken according to c^2, the current neighbors $\mathcal{N}_j$ are assumed to remain static. Therefore if only the considered neighbours must remain static, it is straightforward to think that some nodes in the set $N \setminus \{\{j\} \cup \mathcal{N}_j\}$ can start their computation at the same time.

When a node accesses the same set of resources whenever it accesses any resource, the problem is a generalized form of the paradigmatic dining philosophers problem [73]. The Scheduling by Edge Reversal procedure is a distributed algorithm that solves the dining philosophers problem guaranteeing starvation- and deadlock-freedom. The SER begins operating in an acyclic orientation of the underlying graph. In such orientation, there will exists nodes called *sink*. The sink nodes are those nodes for which all oriented edges are directed to them. Once the sink nodes use the shared resource, they reverse the orientation of the incident edges. For doing so, another set of sink nodes is created and the process goes on guaranteeing that all nodes have access to the resources at some finite time.

Every node on the digraph will become a sink after a certain number of reversals. This number is related with the longest path from the node to a sink. This number is usually called Λ(This number is actually called λ, but in order to avoid confusion with the algebraic connectivity, we decided to use the capital lambda). With all Λ calculated, it is possible to split the nodes in Λ-layers. A plot example of the Λ-layers decompositions can be seen on Figure 5.5.

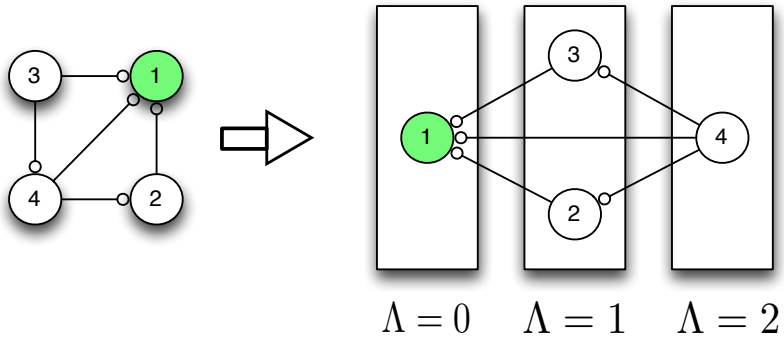


Figure 5.5: Example of a Λ-layers decomposition [8] (courtesy of Daniel Alves).

In the SER mechanism, the first orientation of the graph defines the degree of parallelism of the system. Unfortunately, generating acyclic orientations that maximize concurrency has been proved to be an NP-Complete problem [74], since it is compared to the graph colouring problem. Therefore, other studies had been conducted towards the generation of better orientations aiming to maximize the concurrency.

In such direction, [75] proposed a randomized algorithm, known as Calabrese/França algorithm, to generate acyclic orientations on generic graphs. The fundamental idea is to define a biased or unbiased dice d_i, associated with a vertex $v_i \in G$. For each round, this dice returns a value 0 or 1, which are randomly generated depending on the probability distribution defined by the dice. Let d_i^k and d_j^k be the dice values of an assortment k. A node n_i is called winner if and only if $d_i^k = 1$ and $d_j^k = 0$, for every $n_j \in \mathcal{N}_{P_{k-1}}(i)$, where $\mathcal{N}_{P_{k-1}}(i)$ is the set of probabilistic neighbours of node i. A probabilistic node $j \in \mathcal{N}_{P_{k-1}}(i)$ is the node for which the direction of the edge (i, j) is still not chosen.

With the basic notions of Calabrese/França algorithm, Arantes *et. al.* proposed the *alg-neighbours alg-colors* and *alg-edges* variants, in order to generate acyclic orientations to anonymous distributed systems. In such systems the processing units do not have identity numbers. In the *alg-neighbours* procedure the nodes throws a dice with $n \geq 2$ faces, and the winner node directs all incident edges to it and stop the orientation process. The process will remain to the other nodes until all edges become oriented. The *alg-colors* procedure uses the same idea, but there is a prior phase of colouring before generating the orientation. In order to colouring a node i, the algorithm runs the trials on dices and the node with a better score chooses the least available colour from the group. For each node whose colour is defined can finally choose the edges incident to other coloured nodes.

It is true that the above mentioned procedures to define acyclic orientations to the graph suppose the nodes has no identification. It is straightforward to think that the if the nodes of the system, in the present case we may suppose robots have a

unique ID, the orientation could be given by using the IDs (as in the leader election procedure). However, we chosen the *alg-colours* as the acyclic orientation procedure because it generates acyclic orientations maximizing the concurrency [76]. Figure 5.6 shows a running example of procedure *alg-colours*.

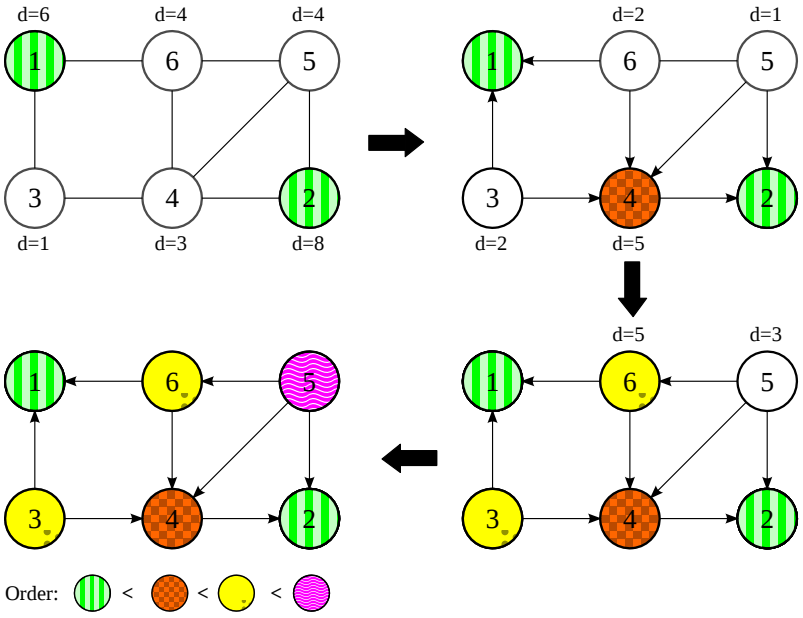


Figure 5.6: A running example of *alg-colors*.

5.3 Final remarks about the computation of c^2

According to [77], the complexity of calculating the determinant of a float matrix is $O(n^4)$. The main advantage of this approach is that the control law that governs the positioning of the relays is independent of the number of robots in the network. This means that the proposed solution is scalable to larger groups of agents, without the necessity of building a more complex control strategy. This scalability feature is a fundamental one when proposing solutions to swarm robotics.

Chapter 6

Heuristic solutions

According to [78], all population-based algorithms produce satisfactory results but there is no such magical heuristic algorithm that, in all optimizing problems, could have a superior performance than others. In order to have a comparison measure of the guidance algorithm proposed in the last chapter, we also have tried out some metaheuristics over the connectivity problem. In this chapter, we show how we have modeled the considered problem for applying some evolutionary algorithms, specifically we chose the Genetic Algorithms (described in section 6.2.1), because it is a classic approach in optimization and a new algorithm called Backtracking Search Algorithm, which is presented in section 6.2.2. Furthermore, we have also evaluated the Standard Particle Swarm Optimization 2011, from a class of standard algorithms proposed by Clerc [79]. The PSO is detailed in section 6.3. The last implemented algorithm is the Simulated Annealing, which is described in section 6.4.

Before starting describing the algorithms, we present the representation of the solution candidates as well as the considered energy function is also approached the next section.

6.1 Introduction

According to Chapter 2, the *one-tiered* version of the connectivity problem is that the active pursuers do not participate of the network as connectivity provider. This does not mean that a same relay can provide connectivity to more than one pursuer. By one hand, allowing pursuers sharing relays could be useful in temporary situations, in other hand this feature could decrease the reachability of pursuers.

The possibility of having any relay to be the connectivity provider of two or more pursuers is not allowed in the model proposed in the Chapter 5. This happens because such relay providers should also considerate its pursuers direction. But when dealing with metaheuristics this can be possible because the relays position

are to be randomly chosen.

In this chapter we also consider that a set of N_p pursuers are trying to reach a set of targets Θ, using a set of relays R to maintain the connectivity. From one second k to $k+1$ the pursuers and relays have to decide over the position in such a way that minimizes an energy function keeping the connectivity. The solutions are depicted in the further sections.

Representation of solution candidates

The considered time is discrete, this means that given the current position of the pursuers and relays, S_k, the problem is to find a solution S_{k+1}, for which the traveled distance of a robot $r^i \in S_k$ and $r^i \in S_{k+1}$ should not be more than a prefixed value R_{step} and the proximity graph induced by the positions in S_{k+1} is a connected one. In this case, for a better data manipulation, each solution can be represented as polar coordinates, i.e. for each robot $i \in P \cup R$, the solution candidate S_{k+1} that includes the positioning of i is a pair with an angle and an absolute value (α_i, p_i), where $\alpha_i \in [0, 2\pi)$ and $p_i \in [0, R^i_{step})$.

Energy function

Using an estimation of the targets positions, the energy function is to minimize the average distance among the pursuers and their assigned targets, in a target-pursuit scenario, for example. But, this metric can be associated with whatever mission the pursuers are engaged with. In this way, the variables involved in the solutions can consider only the relays positioning. After taking a new solution for the relays, one approach could be to move the pursuers deterministically. If the generated solution leads to the graph disconnection, then the energy function may go to infinity value, for example. Therefore, solutions with infinity energy functions are infeasible ones.

Using the same scenario of target-pursuit, we propose the following energy function:

$$J(\mathbf{x}) = w_1 \times \frac{1}{|P_{PA}|} \sum_{i \in P_{PA}, j \in \Theta} \Gamma_{ij} ||x_i - \theta_j||_2 + w_2 \times ConnMeasure(g(\mathbf{x})) \qquad (6.1)$$

where P_{PA} is the set of active pursuers, $g(\mathbf{x})$ is the proximity graph induced by the swarm position $\mathbf{x}$, Θ is the set of active targets, x_l is the state of agent l (in the current case is the 2D position), θ_j is the state of target j, Γ_{ij} is a binary function which is 1 when pursuer i is assigned to target j, and $ConnMeasure(\circ)$ is some connectivity measure over a given graph. Besides, w_1 and w_2 are positive weights.

The function $ConnMeasure(x)$ may be designed in such way that the function could go to infinity when a solution is infeasible as well as it returns some connectivity measure (which can be the number of connected components or some other function of connectivity as exposed on Chapter 2).

Looking at the objective function (equation 6.1) as the average traveled distance, it can be seen that a deadlock situation can be reached when the number of relays does not allow the pursuers to reach their targets. This situation is illustrated in Figure 6.1. In cases like that, the pursuit of some targets should be abandoned or the connectivity of the whole swarm should be relaxed. In scenarios like the connectivity is temporary (where there is only the exchange of information is needed), some targets could be reached, the information could be transmitted and the pursuers could be redirected to non-covered targets in order to deliver such information. Or in scenarios where the targets should be eliminated somehow, the swarm could eliminate the reachable targets and try to pursuit other ones. Such situations are covered and discussed in chapter 7.

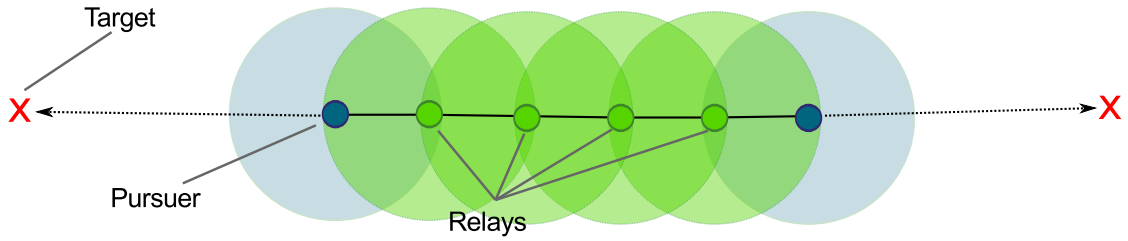


Figure 6.1: Situation where a target pursuit must be paused in order to improve the energy function.

6.2 Evolutionary Algorithms

According to [80], the basis of any evolutionary system is formed by two fundamental forces: a) variation operators (recombination and mutation) which create the necessary diversity and thereby facilitate novelty; and b) selection which acts as a force improving quality. The combination of random changes with fitness-based selection usually enables a search system to evolve solutions faster than random search [81] *apud* [82].

6.2.1 Genetic Algorithms

Genetic Algorithm (GA) is a global optimizer and adaptive strategy procedure [83]. This technique is inspired by population genetics, solutions are chromosomes for which evolution mechanisms such as recombination and mutation are used to reconstruct new solution candidates and evolute the whole population of

chromosomes. There are countless successful applications of GA in the optimization field. The general structure of this technique is described in algorithm 6.2.1.

Algorithm 6.2.1 *General Structure of a Genetic Algorithm [80]*

1: **Generate** Initial Population with random solutions
2: **Evaluate** each candidate
3: **repeat**
4: **Select** parents
5: **Recombine** pairs of parents
6: **Mutate** the resulting offspring
7: **Evaluate** the new candidates
8: **Select** individuals for the next generation
9: **until** *Some stop criteria is satisfied*

Select operator

The **Select** operator is responsible for choosing the parents to be used in the recombination. The *fitness proportional selection* is a method that defines the probability of an individual f_i is selected for mating is $f_i / \sum_{j \in Pop} f_j$. Unfortunately, there are some problems with this mechanism, such as premature convergence through the multiplication of elements who are much better than the rest of the population, loose of selection pressure when the fitness values are very close together, and the differences in the definition of probabilities when dealing with transposed versions of the same fitness function [80].

Therefore, in [84] it is proposed the Goldbarg's sigma scaling which considers the mean $\bar{f}$ and standard deviation σ_f of fitnesses in the population, as illustrated in following equation:

$$\max(f(x) - (\bar{f} - c - \sigma_f), 0.0) \tag{6.2}$$

where c is a constant usually set to 2.

Another way of doing selection is known as *ranking selection*. In such operator the individuals are ranked based on their fitness function and the probabilities are chosen according to a linear of exponential function. The usual formula for calculating the selection probabilities using the linear ranking scheme uses a parameter $s \in (1.0, 2.0]$. The ranking selection is then stated as follows:

$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)} \tag{6.3}$$

where μ is the population size.

It is important to highlight that using this scheme the selection pressure is limited. If more emphasis on selecting individuals with fitness above the average is needed, an *exponential ranking* scheme may be used as shown by the formula:

$$P_{exp-rank}(i) = \frac{1 - e^{-i}}{c} \tag{6.4}$$

where c is a normalization factor in order to keep the sum of probabilities equals the unitary value.

After defining which selection probabilities to be used on selection of parents, the selecting is done by sampling individuals from the population using the defined probabilities. The simplest way of sampling is known as the *roulette wheel* algorithm. Although its simplicity, according to [80], it has been recognized that the roulette wheel algorithm does not in fact give a particularly good sample of the required distribution. Therefore, whenever more than one sample needs to be picked up from the distribution, the use of an algorithm called *Stochastic Universal Sampling* (SUS) is preferred.

Assuming sampling λ members out of μ parents, first a list of values $[a_0, a_1, \ldots, a_{\mu-1}]$ such that $a_i = \sum_0^i P_{sel}(i)$, where P_{sel} is the selection distribution (proportional or ranking). The Roulette and Stochastic Universal algorithms are shown by algorithms 6.2.2 and 6.2.3, respectively.

Algorithm 6.2.2 *Roulette wheel algorithm [80].*

```
1: currentmember = 1
2: while currentmember ≤ μ do
3:     r = U(0, 1)
4:     i = 1
5:     while a_i ≤ r do
6:         i = i + 1
7:     end while
8:     matingpool_currentmember = parents_i
9:     currentmember = currentmember+1
10: end while
```

Algorithm 6.2.3 *Stochastic Universal Sampling algorithm [80].*

```
1: currentmember = i = 1
2: r = U(0, 1/μ)
3: while currentmember ≤ μ do
4:     while r ≤ a_i do
5:         matingpool_currentmember = parents_i
6:         r = r + 1/μ
7:         currentmember = currentmember+1
8:     end while
9:     i = i + 1
10: end while
```

The last selection procedure presented here is the *Tournament*. In such procedure a number $k \geq 2$ of individuals are randomly picked up and using a probability of p (usually equals to 1.0) for the fittest to win the tournament.

Recombination

When dealing with float-point strings, recombination operators can usually be applied with two ways: *discrete recombination* and *arithmetic recombination*. In the former, the gene string is split at a random point and resulting offspring comes from combining the split parts. The latter consists of updating each allele with a weighted sum of the parent's alleles .

A simple recombination consists of picking a recombination point k, then the first child has the first k floats of parent 1 and the rest is arithmetic averaged of parent 1 and 2. The second child is produced in the analogous way. Another way is to apply single arithmetic recombination which consists of picking a random allele k and averaging its value with the parents'. Another surrogate way is taking the weighted average value of parents' alleles ($z_i = \alpha x_i + (1-\alpha)x_j$, for parents i and j).

About the disadvantages of discrete recombination is that there is no production of new genetic material, while for the arithmetic recombination the offspring will always be in the average value of its parents.

Mutation

For floating-point representations, the common way for applying mutation is to update alleles values of each gene randomly with a new value picked up from the domain. Normally used with a position-wise mutation probability, the uniform mutation takes values for the allele using the uniform distribution. Besides, there are non-uniform mutations with fixed distribution. However this does not imply

that some other distributions can be used. For small changes around the current value, for example, the Gaussian or Normal distribution could be employed. As well as an alternative is to use the Cauchy distribution, which has a "fatter" tail, and therefore the probabilities of generating modifications with larger size are slightly higher than for the Normal using the same standard deviation.

Survivor Selection

The survivor selection is the procedure of choosing who of the μ parents and λ offspring will compose the next generation. *Replace worst* is a fitness-based scheme which takes the worst λ members of the population for the replacement. This strategy is usually employed when the population has a large number of individuals in conjunction which a "no duplicates" policy, because it can lead to premature convergence.

Another known scheme is *elitism* which uses a conjunction of age-based and stochastic fitness-based replacement schemes. The fittest is always kept in the next generation. If it is chosen to be replaced and none of the members in the offspring has a better or equal fitness value, it is kept for the next generation.

6.2.2 Backtracking Search Algorithm (BSA)

The Backtracking Search Algorithm (BSA) is an evolutionary algorithm (EA) that was introduced by Civicioglu in 2013 [85]. The BSA has a single control parameter (the mixture rate, which will be explained later) and includes a simple structure which can be shown in algorithm 6.2.4. In addition, the algorithm presents two new crossover and mutation operators. Another difference is in the memory of a population, it stores a population from a randomly chosen previous generation that is used for generating the search-direction matrix.

Algorithm 6.2.4 *General Structure of BSA [85]*

```
Initialization
repeat
    Selection-I
    Begin Generation of Trial-Population
        Mutation
        Crossover
    End
    Selection-II
until Some stop criteria is satisfied
```

The *Initialization* step consists in initialize an initial matrix $P_{ij} = rand(low_j, up_j)$, $i = 0, \ldots, N-1$ and $j = 0, \ldots, D-1$, where N is the population size, D is the dimension of the solution, $rand(\cdot)$ is the uniform random generate function, and *low* and *up* are the lower and upper bound vectors for the numerical entries of the solution candidate P_i.

The *Selection-I* stage determines the historical population ($oldP$) to be used for calculating the search direction. The very first historical population is generated as the same procedure described in the *Initialization* step. For the other ones, a population P is chosen to be the $oldP$ if $rand(0,1) < rand(0,1)$. This memory population is hold until the previous criteria is satisfied again, and then the $oldP$ is updated. After $oldP$ is determined, the $oldP$ receives an operation to shuffle the order of the individuals. The indexes are chosen by randomly generating a permuting vector of their position.

$$Mutant = P + F.(oldP - P) \tag{6.5}$$

The *Mutation* operator is defined by equation 6.5. The parameter F controls the amplitude of the search-direction matrix ($oldP - P$). It can be seen that the search-direction includes the experience of previous populations. In [85], the parameter F is given by the formula $F = 3.randn(0,1)$, where $randn(0,1)$ generates numbers according to the standard normal distribution.

The *Crossover* operator takes two parameters the trial population resulting of the *Mutation* step and a *mix rate* value. The crossover procedure is shown in Algorithm 6.2.5. The *mix rate* parameter controls the amount of elements of individuals that will mutate in a trial. This operator is composed of two predefined strategies to randomly define the BSA's map. The first strategy uses the *mix rate* parameter (algorithm 6.2.5, lines 4–7). The second strategy allows only one randomly chosen individual's element to mutate in each trial.

Algorithm 6.2.5 *BSA's crossover operator [85]*

procedure BSACrossover($T, mixrate$)
  Let T be the trial population after the mutation stage and $mixrate$ a value that controls the amount of elements of individuals that will mutate in a trial.
  $map_{(0:N-1,0:D-1)} = 1$ ▷ map is a matrix of $N \times D$ ones.
  if $rand(0,1) < rand(0,1)$ **then**
    for i = 0, ..., N-1 **do**
      $map_{i,u_{0:\lceil mixrate.rand.D \rceil}} = 0$ ▷ $u = permuting(\langle 1, 2, \ldots, D-1 \rangle)$
    end for
  else
    for $i = 0, \ldots, N-1$ **do**
      $map_{i,randi(D-1)} = 0$ ▷ $randi(\cdot)$ generates random integers
    end for
  end if
  for $i = 0, \ldots, N-1$ **do**
    for $j = 0, \ldots, D-1$ **do**
      if $map_{ij} == 1$ **then**
        $T_{i,j} = P_{i,j}$
      end if
    end for
  end for
  return T
end procedure

In order to control the boundary of the elements generated by the two above mentioned phases, the BSA algorithm also includes a boundary control procedure. For each individual in the population, this procedure verifies the values of each variable belonging to the domain. If the value overflows the allowed search-space limits, then the variable is substituted by a randomly generated one.

The last step of the BSA is the *Selection-II*. In this phase, the T_i's with better fitness values than the corresponding P_i's are used to update the P_i's. A variable P_{best} is used to store the best individual across the epochs.

6.3 Particle Swarm Optimization

Originally proposed in [86], the Particle Swarm Optimization is inspired on social behaviour and interaction among the members of a swarm (flock of birds, colony of bees, for example). The group behaviour is influenced by the combined experience of a single particle as well as the whole group. According to [87], the main inspiration of

PSO comes from the simulation and analysis of social dynamics and the interactions among the members of organized colonies, therefore, it is categorized as a swarm intelligence algorithm.

Moreover, PSO is a versatile population-based optimization technique, similar to evolutionary algorithms, but with no crossover nor mutation operators. Basically, particles "fly" above the fitness landscape, while a particle's movement is influenced by its attraction to its neighbourhood best (the best solution found by members of the particle's social network), and its personal best (the best solution the particle has found so far) [12].

In PSO, the population is formed by individuals called particles. For each particle, there are two main properties: the particle dynamics and the particle information network. In such algorithm, particles move over the search space using the following equations:

$$v(t+1) = v(t) + a(t+1) \tag{6.6}$$

$$x(t+1) = x(t) + v(t+1) \tag{6.7}$$

where a, v, x and t are acceleration, velocity, position and time, respectively.

Each particle's acceleration parameter is updated following two attraction forces. The first one is the local best (indicated by *pbest*), which is a memory of the best positioning reached by the particle over time. Another factor composing the attraction force is the global best (or the neighbourhood best, indicated by *gbest*). In this case, some network topologies have been tried in some research in order to find out the neighbourhood impact over a particle in searching for solutions. According to [12], the fully connected network is a popular choice for unimodal problems and therefore there is a single *gbest* representing the best location found by the whole swarm.

There is a class of PSO called Standard PSO (SPSO for short). There are three main versions of such standard which are the SPSO 2006 [79], the SPSO 2007 [11] and SPSO2011 [88]. Regardless the changed aspects, they follow a canonical basis, which are depicted in Algorithm 6.3.1. The Table 6.1, summarizes the main differences among them. About the velocity update, the SPSO2011 uses the following equation to update the velocity:

$$v_i(t+1) = wv_i(t) + x_i^{'}(t) - x_i(t) \tag{6.8}$$

where $x_i^{'}$ is a random point (not necessarily uniformly picked) that lies in the hypersphere $H_i(G_i, ||G_i - x_i||)$ for which $G_i = x_i + c\frac{p_i + l_i - 2x_i}{3}$

Table 6.1: Comparison among Standard PSO implementations [11]. The function $[u]$ gives the integer part of u.

Feature	*SPSO2006*	*SPSO2007*	*SPSO2011*
Swarm Size	$10 + [2\sqrt{D}]$		User defined (40 is a suggested number)
Initialization	$\begin{cases} x_i(0) = U(min_d, max_d) \\ v_i(0) = \frac{U(min_d, max_d) - x_i(0)}{2} \\ p_i(0) = x_i(0) \\ l_i(0) = \arg\min_{j \in N_i(0)}(f_i(p_j(0))) \end{cases}$		$v_i(0) = U(min_d - x_{i,d}(0), max_d - d_{i,d})$
Velocity Update	$\begin{cases} v_{i,d}(t+1) = wv_{i,d}(t) + \\ \qquad U(0,c)(p_i(t) - x_i(t)) + \\ \qquad U(0,c)(l_i(t) - x_i(t)) \end{cases}$		Equation 6.8
When local best= previous best	Nothing special	$v_{i,d}(t+1) = wv_{i,d}(t) + U(0,c)(p_i(t) - x_i(t))$	$G_i(t) = x_i + c\frac{p_i - x_i}{2}$
Confinement	$\begin{cases} \text{if } x_{i,d}(t+1) < min_d \text{ then } \begin{cases} x_{i,d}(t+1) = min_d \\ v_{i,d}(t+1) = 0 \end{cases} \\ \text{if } x_{i,d}(t+1) > max_d \text{ then } \begin{cases} x_{i,d}(t+1) = max_d \\ v_{i,d}(t+1) = 0 \end{cases} \end{cases}$		Similar to 2007 but $v_{i,d}(t+1) = -0.5v_{i,d}(t+1)$

Algorithm 6.3.1 *Canonical PSO [12]*

for each particle i **do**
    Random choose $v_i, x_i = p_i$
    Evaluate $f(p_i)$
    $g = \arg\max(f(p_i))$
end for
repeat
    for each particle i **do**
        Update v_i, x_i
        Evaluate $f(x_i)$
        if $f(x_i) > f(p_i)$ **then** ▷ Updating the local best
            $p_i = x_i$
        end if
        if $f(x_i) > f(p_g)$ **then** ▷ Updating the global best
            $p_g = \arg\max f(p_i)$
        end if
    end for
until Termination criteria reached

6.4 Simulated Annealing Approach

The *Simulated Annealing* (SA) has emerged from some works developed by S. Kirkpatrick *et al* [89] and V. Cerny [90]. SA is a meta-heuristic process of local search that accepts not improved solutions with a certain probability in order to escape from local optimal points. This process is based on a procedure used to annealing material to a maximal equilibrium state. Such maximal equilibrium is the "cooling" of a problem to be optimized, starting at a high temperature and then slowing down such temperature until no changing can be reached [89].

The SA algorithm begins the search from any initial solution. Such initial solution (S_0) can be chosen by heuristic methods or stochastic ones. For each iteration, a neighbor solution (S') is generated from the current one (S). Neighbor solutions that improve the result of an energy cost function are always accepted. Worse neighbor solutions can be accepted with a certain probability (Boltzmann Probability Density function is used in general). As the temperature (T) is decreasing, the probability of acceptance of such worse neighbor solution decreases as well. Considering the minimization case, Let $\Delta = J(S) - J(S')$, then the probability of a worse solution S' be accepted ($S \leftarrow S'$), over a temperature T is expressed as [91]:

$$P_T(S \leftarrow S' | \Delta < 0) = e^{-\frac{\Delta}{T}} \tag{6.9}$$

$$Pr(S_m \in \mathbf{S}^*) \to 1 \quad as \quad m \to \infty \tag{6.10}$$

where $\mathbf{S}^*$ is the global optimal solution set and S_m is the solution S on step m, under the cooling scale defined by:

$$T_k = \frac{\Phi}{\log k} \tag{6.11}$$

where Φ is a constant proportional to the value of J.

Under these circumstances, the algorithm will converge to a global optimum, with certain probability, if and only if, when time $k \to \infty$ [92]. The whole algorithm can be seen in Algorithm 6.4.1.

When dealing with meta-heuristics, a common questions are how to implement the neighbourhood generation function, how to define the objective function and how to generate the initial solution. In SA, the main project issues are:

- The probability function of accepting worse candidate solutions.
- The cooling rate: this is a fundamental aspect for which has a fundamental role in the speed and quality of solutions.

Algorithm 6.4.1 *Simulated Annealing Algorithm*

```
1: function SIMULATEDANNEALING(S)
2:     while T > T_min do
3:         repeat
4:             Generate a neighbor solution S' from S
5:             Δ ← J(S) - J(S')
6:             S* ← arg min{J(S), j(S'), J(S*)}
7:             if Δ < 0 then
8:                 S ← S' with probability e^{-Δ/T}
9:             else
10:                S ← S'
11:            end if
12:        until Some iteration criteria is reached
13:        Update T
14:    end while
15:    return S*
16: end function
```

Generating neighbor solutions

As could be seen in the model, each solution candidate should fulfill two restrictions:

1. The step size should be less than or equal to a given value ("movement constraint").

2. After the movements, the swarm should stay connected ("connectivity constraint").

We call the first one as the *movement constraint* and the second one as the *connectivity constraint*. In order to generating only neighbor solutions for which satisfies (1), we decided to random an angle which varies from 0 to 2π and an uniformly generated variable from 0 to 1 that multiplies the maximum distance allowed from the point in S^0 to whichever S'. In Figure 6.2, it is illustrated S^1 being a neighbor solution of S^0, as well as $S^2 \in N(S^1)$. Whichever be the solution S', it should respect the movement constraint ($\leq D_{max} = R_{step}$).

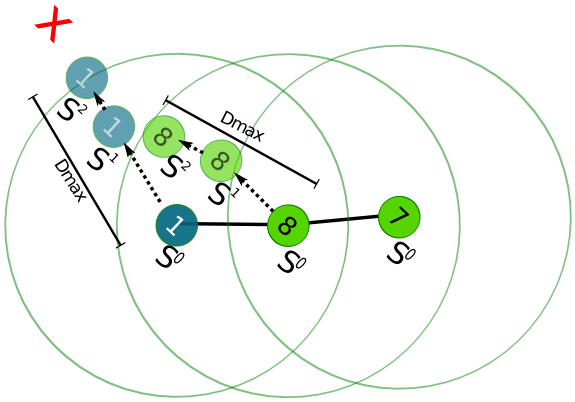


Figure 6.2: Situation where from S^0 to S^2 we have the amount of traveled restriction limited by the configuration value D_{max}.

The connectivity constraint is not explicitly handled due to allow another reachable network configurations. If we explicitly constraint the step size to be in the connectivity radius, the algorithm could not be able to find other connected configurations. By the other hand, by generating solutions only considering the step size constraint could lead to a high number of unfeasible solutions (disconnected configurations).

6.5 Final remarks on the considered solutions

In this chapter we have presented some of the chosen heuristic solutions, as well as we depicted how to represent the solutions and the energy function. One natural question is why we have chosen the aforementioned techniques. This choice was guided as we would like to test some of the traditional solutions (such as Genetic Algorithms and Simulated Annealing) as well as to try new ones (Backtracking

Search Algorithm and Standard PSO 2011). In the next chapter we present our methodology and some experimental results over the proposed solutions.

Chapter 7

Experimental Results

In this chapter it is presented the test methodology as well as the results of the afore proposed solutions.

7.1 Review on testing scenarios

Before presenting the chosen evaluation methodology, it is important to review the ways researchers have been using to present their results. This is necessary, because there is some limitations on the fixed images when there is the necessity of showing the performance of moving agents. Therefore, in the next paragraphs it is mentioned how the results are presented in the correlated literature.

In [38], the authors present their results from a setup of six robots positioned in a line formation on the plane. The objective is to position the robot in order to increase the algebraic connectivity. So in the same graph, all trajectories points are plotted highlighting the initial and final state of the agents. Moreover, in [**?**], a setup of 9 agents, where one of them plays a role of leader with a proper trajectory, and the other 6 should follow it without loosing the group connectivity. The results are in general shown using screenshots of some seconds of simulations including the initial and final states.

In [60], the authors simulated the connectivity maintaining algorithm over a randomly generated six-node network. In such scenario, three robots played the role of leader implementing the following motion model: $\dot{p}_x^i = -0.2$, $\dot{p}_y^i = 0.5cos(p_x^i)$. The other three followed the control law proposed by the authors, which consisted of increasing an estimated algebraic connectivity λ_2. Again, some screen-shots of some seconds of simulation had been taken and shown.

In [29], the authors simulated a swarm of 30 robots executing two tasks: rendezvous and formation preserving. The former is to drive the robots to a common, not a priori specified location without relying on global positioning. While the latter consists of driving the robots from an initial connected setup to a desired

global formation (from a random form to a circle, for example). Furthermore, in [93], a formation of six agents is controlled to reach a target position. They ran each simulation 30 times, and for each setup the robots were deployed in random formation with the restriction of the barycenter of swarm was 15m far from the objective point. As a quality measure, they accounted the average control effort. They computed the control effort as the sum of the norm of the input control vector.

7.2 Evaluation Methodology

The first hypothesis is that there is an assignment between *targets* and *active pursuers*. *Active pursuers* is the label to pursuer agents that are following its mission instructions. The considered application scenario assignees a mission of trying to capture all targets in the environment, i. e., to minimize the average sum of squared distance between the set of targets and the set of active agents. At the very first second, as the vehicles are deployed very closely to each other, this can be done using a small set of messages, because the initial proximity graph is a complete one. Therefore, it is assumed that for each second the assignment is already known based on the last estimation of targets' positions. For the simulations exposed on this chapter, the Hungarian method [50] has been used to assign pursuers to targets, but only to give a direction for the pursuers. The Hungarian method is shown on algorithm 7.2.1.

Algorithm 7.2.1 *Hungarian method for assignment*

1: Input: $A_{n\times n}$ cost matrix
2: $A[i:] \leftarrow A[i:] - \min_{a_j \in A[i:]} a_j$ ▷ From each row, subtract min element from it
3: $A[:i] \leftarrow A[:i] - \min_{a_j \in A[:i]} a_j$ ▷ From each col, subtract min element from it
4: Let nl be the minimum number drawn lines covering all entries equals to 0 in A
5: **if** $nl = n$ **then**
6: Finished: An optimal assignment of zeros has been found
7: **else**
8: Let x be the smallest $a_{ij} \in A$ such that is not covered by any line
9: Update $A \leftarrow A - x$
10: Go to step 4
11: **end if**

Algorithm 7.2.2 Abandon farthest target policy.

1: **if** $t > 0$ **and** $|ObjF(t-1) - ObjF(t)| < \epsilon$ **then**
2: farthestPursuer $\leftarrow$ findFarthestPursuer(pursuers)
3: $pursuers[farthestPursuer].role \leftarrow$ **passivePursuer**
4: **end if**
5: **if** *TargetList*.size has decreased **then**
6: **for** $p \in passivePursuers$ **do**
7: $p.role \leftarrow$ **pursuer**
8: **end for**
9: **end if**

When the network is stretching in order to track the assigned targets, the underlying graph can achieve its maximum reachability. In such case, we applied a *Abandon farthest target policy*. Such policy (described on algorithm 7.2.2) is activated when the objective function does not improve for a period of time (in the present simulations the horizon is 1). Once this happens, the farthest active pursuer changes its role to *passive pursuer*. The mission of a *passive pursuer* is to follow its relay provider. We have tried to change the role of an *active pursuer* to a temporary relay, a role known as *pursuer relay*. Unfortunately, when doing so, the *pursuer relay* engages into the relay network creating connectivity dependencies. If we had a hybrid model for which any robot could play any role, then this would not be a problem, since the leaves in the graph could naturally become pursuers. In this case, only a new assignment would be necessary between the leaves of the underlying graph and the remaining of targets. Figure 7.1 illustrates this problem. When the number of searched targets decreases, all passive pursuers may become active ones, and consequently they run for the assignment again.

It is evaluated a scenario where the pursuers are trying to reach their assigned targets and "capture" them. The "capture" of a target is understood as a pursuer reaching its position. When this happens the target disappears from the scenario.

Over this scenario, the following metrics had been taken:

1. Objective function value (in this case the quadratic distance among pursuers and targets) over time
2. Success Rate (number of captured targets over time)
3. Processing time
4. Maximum number of simulated seconds to reach the final state (all targets being captured or the expiration time being reached)

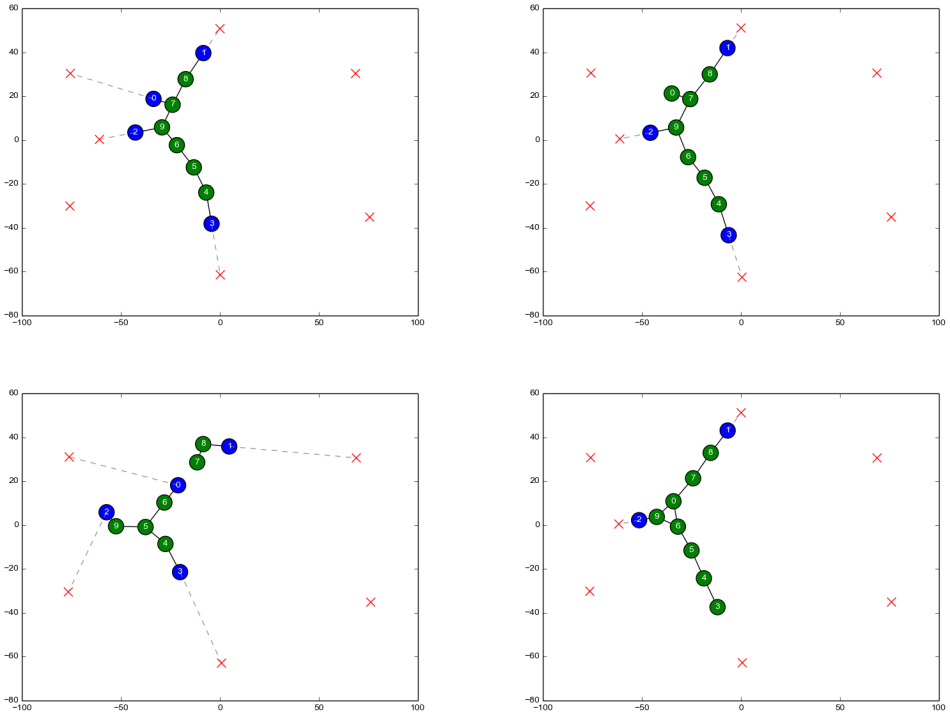


Figure 7.1: Illustration of the problem of a *relay pursuer* coming back to the role of *active pursuer*. The sequence is clockwise. The pursuer 0 is asked to become a relay, but as it engages the connectivity network, when it need to come back to the pursuer role, it implicates on connectivity break.

Computational Tools and Setup

For the experiments exposed on this chapter, the considered language is Python 2.7.6 version, with the `numpy` package, version 1.8.2, for the linear algebra calculations. In addition, the `matplotlib` package, version 1.3.1, it was used for plotting the graphics used in this document. All tests ran over a 64 bit Linux kernel version 3.13.0-37, running on a Intel i5 1st generation, 1.2GHz, with 4GB of memory. All time measures had been taken using the python package `time`. We also used a python package called `munkres` which contains an implementation of the Hungarian method used for assignment tasks.

7.3 Test instances

In the correlated research, unfortunately it was not be found (until now) some benchmark set of instances in order to compare the algorithms. As could be seen in Chapter 3, the maintenance connectivity problem has been addressed in so many different ways as well as for distinct purposes.

In order to evaluate the proposed solutions, some instance problems have been suggested as well. In each instance, there is an initial group of targets which are positioned in a 2D space area. Each agent is guided with a predefined behavior. From the literature, we have identified six different types of trajectory behavior, which it is described in Table 7.1.

Table 7.1: Target behaviors.

	behavior	**Description**
1	Escape	The targets try to escape from the closest pursuer.
2	Static	It remains in the same position during all simulation.
3	Spiral	It starts a spiral trajectory from the deployed start point.
4	Cooperative	Try to move towards its closest pursuer.
5	Random	Random movements.

For evaluating the proposed algorithms, we introduce two scenarios named $allLeft$ and *symmetric*. In the $allLeft$ scenario a set of 7 targets are deployed in the left of the initial point of *relays* and *pursuers*. This can be seen in Figure 7.2. Similarly, in scenario *symmetric* 8 targets are symmetrically deployed around the initial point of *relays* and *pursuers*. This last scenario can be seen in Figure 7.3.

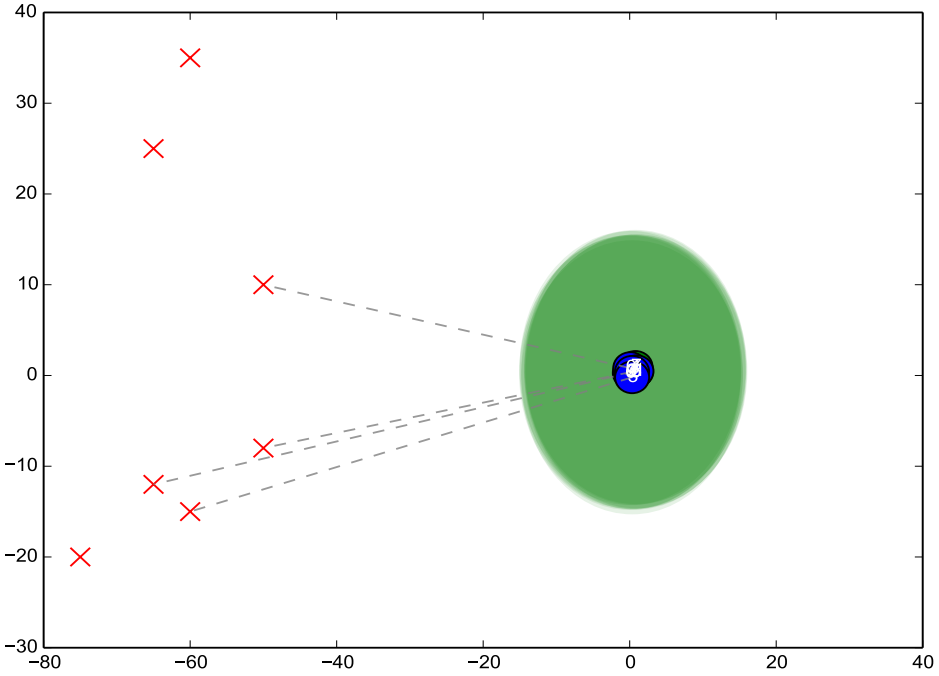


Figure 7.2: Scenario *allLeft* where the seven targets are on the left of the initial deployment of the swarm.

Table 7.2: behavior schemes.

behavior Scheme	**Description**
allStatic	All targets remain where they are initially deployed.
allRandom	At each moving second, all targets pick a random angle and velocity
4Evasive4Static	Four targets escapes from its closest pursuer. The remaining stay static.
4Collaborative4Spiral	Four targets are trying to move towards their closest pursuers while the remaining are moving in spiral movements.

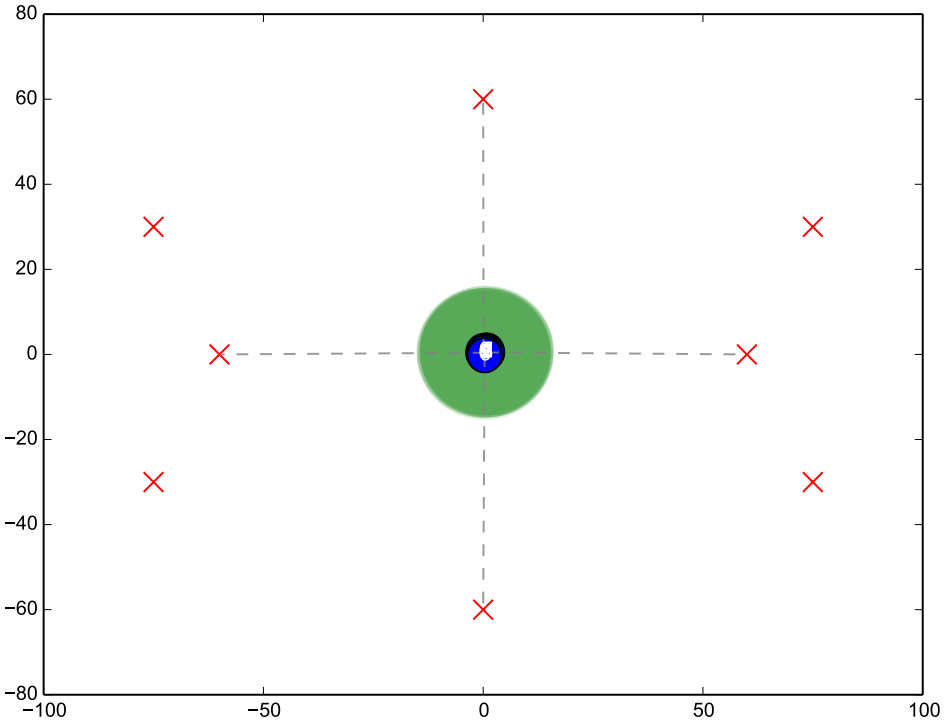


Figure 7.3: Scenario *symmetric* where eight targets are initially deployed in a symmetric way around the initial deployment of the vehicles

Once the scenarios have been set, we also propose four behavior schemes which are depicted on Table 7.2. The all static is interesting case when dealing with fixed targets for which the swarm should reach. For example, the capture state could seen as deliver some resources to fixed spots in a map, for example. The *allRandom* case illustrate the undefined routes of these moving targets. The other combinations are randomly chosen. The *escape* and *spiral* behaviors are the most difficult to handle. All *relays* and *targets* have a fixed maximum velocity of $5m/s$, with a communication range of $15m$. The weighting function was the one as the Figure 2.6, with $\rho_1 = 4$ and $\rho_2 = 7$. The *target*'s maximum velocity was set to $2m/s$. The results are presented in the next section.

7.4 Results based on the test instances

We evaluate the *SwarmLambda2* solution with and without the SER variation. A good property for the solutions is to find the set of positions using the minimum amount of processing. For that reason, we have chosen small populations for the stochastic solutions. For each such solution, we evaluate three distinct parameter sets, as shown by Tables 7.3, 7.5, 7.4 and 7.6. For each environment × behavior

scheme × solution setting we have ran 8 trials. Tables A.1,A.2,A.3 and A.4 summarizes the results for the environment *allLeft*, as well as Tables A.5, A.6, A.7 and A.8 the results for the environment *symmetric*. In order to simplify the presentation of the results, we show the best results in tables .

Table 7.3: BSA settings.

#	**Pop. Size**	**Dim. Rate**	**# of generations**
1	50	0.2	50
2	30	0.2	50
3	10	0.2	20

Table 7.4: GA settings.

#	**Pop. Size**	**CrossOver**	**Mutation**	**# of generations**
1	50	0.1	0.05	10
2	20	0.1	0.05	20
3	10	0.1	0.05	30

Table 7.5: SA settings.

#	**High Temperature**	**Freeze Temperature**	**# setps at each temp.**
1	2.0	0.5	20
2	2.0	0.5	10
3	2.0	0.5	5

Table 7.6: PSO2011 settings.

#	**Pop. Size**	**# of generations**
1	50	10
2	30	8
3	10	5

The boldfaced metaheuristics configurations indicate the best performance in terms of **processing time** as well as the sign * indicate the best metaheuristic solutions in terms of *max simulated seconds* to reach the final state (capture all

targets). We established a total of 200 seconds for the max of simulated seconds, i.e, if a solution spent 200 seconds and the success rate is less than 1.0, it means that the vehicles finalized their mission without capturing all targets. SwL_2 and SwL_2(SER) stands for *SwarmLambda2* and *SwarmLambda2* with SER mechanism for ordering the processing of each unit. The subscript number under the metaheuristics indicate which configuration has been reported according to the settings aforementioned.

Scenario *allLeft*

Table 7.7: Scenario *allLeft* behavior *allStatic*.

	SwL_2	SwL_2(SER)	SA_1^*	BSA_1^*	**PSO$_2$**	**BSA$_2$**
Avg. Proc. Time	0.11	~ 0.23	1.50	1.74	0.08	0.19
Success	1.0	1.0	1.0	1.0	0.9	0.7
S. Seconds	23	23	22	19	200	200

Table 7.8: Scenario *allLeft* behavior *allRandom*.

	SwL_2	SwL_2(SER)	BSA_1^*	SA_1^*	**GA$_3$**	**SA$_3$**
Avg. Proc. Time	~ 0.12	~ 0.45	~ 1.50	~ 1.31	~ 0.16	~ 0.45
Success	1.0	1.0	1.0	1.0	0.4	0.9
S. Seconds	~ 25	~ 28	~ 24	~ 32	200	200

Table 7.9: Scenario *allLeft* behavior *4Evasive4Static*.

	SwL_2	SwL_2(SER)	SA_1^*	SA_1^*	**PSO$_2$**	**SA$_3$**
Avg. Proc. Time	0.11	~ 0.65	1.5	1.74	0.08	0.41
Success	1.0	1.0	1.0	1.0	0.9	0.7
S. Seconds	24	24	22	19	200	200

Table 7.10: Scenario *allLeft* behavior *4Collaborative4Spiral*.

	SwL_2	SwL_2(SER)	BSA_3^*	SA_1^*	**GA$_2$**	**GA$_3$**
Avg. Proc. Time	0.14	~ 0.17	0.94	1.62	0.16	0.17
Success	1.0	1.0	1.0	1.0	1.0	1.0
S. Seconds	23	23	15	16	52	58

Scenario *symmetric*

Table 7.11: Scenario *symmetric* behavior *allStatic*. There were not metaheuristics with success rate greater than 0.5.

	SwL_2	SwL_2(SER)
Avg. Proc. Time	0.07	~ 0.45
Success	1.0	1.0
S. Seconds	199	143

Table 7.12: Scenario *symmetric* behavior *allRandom*. There were not metaheuristics with success rate greater than 0.5.

	SwL_2	SwL_2(SER)
Avg. Proc. Time	0.06	~ 0.45
Success	3 cases with 1.0	1 case with 1.0
S. Seconds	150 (best)	151

Table 7.13: Scenario *symmetric* behavior *4Evasive4Static*. There were not metaheuristics with success rate greater than 0.5.

	SwL_2	SwL_2(SER)
Avg. Proc. Time	~ 0.07	~ 0.68
Success	1.0	1.0
S. Seconds	143	158

Table 7.14: Scenario *symmetric* behavior *4Collaborative4Spiral*. There were not metaheuristics with success rate greater than 0.5.

	SwL_2	SwL_2(SER)
Avg. Proc. Time	0.06	~ 0.55
Success	1.0	1.0
S. Seconds	71	70

7.5 Final remarks on the general results

Generally speaking, the heuristics are a great solution in case of whatever weighting function could be used. In the *SwarmLambda2* solution, the weighting function should necessarily be a decrease function of the distance between agents. Another difference of using the heuristics is that the one-tier is still maintained and a relay can be a connectivity provider of more than one pursuer. While in the *SwarmLambda2* solution a pursuer needs an exclusive relay provider. Of course another assignment among relay-provider and pursuers could be provided over time. This was not tested because it is believed that as the pursuers and relays are initially deployed close to each other, and because of the model, the initial relay provider would be the fittest one to maintain as its role. However, further investigations towards such dynamic assignment could generate better results.

Using the proposed model to model the energy function used in the heuristics, it could be observed that when the communication radius is reasonably greater than the allowed step size, the initial solutions on the relays mode leads to the same energy amount. This is due to the fact that the connectivity measure used in such function is used only to classify the solution as feasible and infeasible. Maybe an investigation towards the use of some connectivity indicator could be used in order to give a hint on good relay positioning for initial solutions.

Despite the adaptability of the heuristics, their processing times, when compared to the *SwarmLambda2* is still a drawback. However, by comparing the *SwarmLambda2* with and without the SER mechanism, it could be observed that the order for which the relays are chosen to decide its movements implicates on the maximum number of seconds spent to reach the final state of the simulation. Therefore, future investigations towards the adaptation of the SER mechanism in order to prioritize the relay-providers as sinks would probably integrate the two advantages of such kind of solution.

Chapter 8

Visual Tracker

This chapter composes the second general objective of this thesis which consists of proposing a realtime tracker of generic objects to improve the vision of the drone. In order to reach such objective, it was necessary to study for pattern recognition techniques for which could contribute to realtime processing. In this way, it was decided to investigate the WiSARD weightless model as a classification system on the problem of tracking multiple objects in realtime. Exploring the structure of this model, the proposed solution applies a re-learning stage in order to avoid interferences caused by background noise or variations in the target shape. So the problem consists of given the bounding box covering the desired object in the first frame, the objective is to adapt to the new shapes of the object following it through the screen view. Moreover, a mid- and short-term memory scheme is proposed in order to overcome problems in past shapes of the followed object.

8.1 Related Work

Tracking objects in real-time is an important and challenging task, useful for many applications. Among its challenges, the real-time requirement is an obstacle for many off-the-shelf tracker solutions due the high cost of processing. Therefore, a fast tracker solution with exploration of the total observed area is decisive for such type of application. The exploration of the whole image area has a high cost, so using parallel approaches sounds an interesting way to achieve it.

SanMiguel et al. [94] proposed a framework for video tracking algorithms quality estimation, which features the capability of evaluating video trackers with multiple failures and recoveries over long sequences. Percini and Del Bimbo [95] presented a tracking method that uses multiple instances of scale invariant local features, and a non parametric learning algorithm based on the transitive matching property, showing state of the art tracking performance on public available benchmark datasets. In [96], the WiSARD model has been successfully used by an artificial

vision system in order to follow the cadence of ships, implying in a model of the movement of an observed vessel.

In this work, a part of [94] is used as the means of evaluating tracker accuracy. Besides, the adopted methodology takes the opposite approach of [95] by adopting a minimum number of features, focusing on portability and speed of the tracker. Moreover, the objective of this paper is evaluate the use of WiSARD neural model, taking advantages of its structure in order to overcome the real-time requirement of an on-line tracker for general objects.

This paper is organized as follows. The WiSARD models is presented in section 8.2. In following, the section 8.5 describes how we used the WiSARD as the classification system of the proposed tracker application. Then, the experimental setup followed by some results are presented in section 8.5.1. Finally, in section 8.6 some conclusions and future improvements are pointed.

8.2 WiSARD

The WISARD is a weightless neural network model conceived, initially, for bill recognition and to be implemented in hardware. WiSARD stands for Wilkie, Stonham and Aleksander's Recognition Device [9]. The model has its neuron unit based in the RAM memory. In training mode, this RAM memory stores "1" on its memory position addressed by the binary input pattern (the non-addressed entries remain "0"). While in classifying mode, the RAM outputs the value addressed by its input. Thus, a RAM fires when the input pattern addresses a value equals to "1".

As an advantage over the McCulloch and Pitts neuron model, the RAM-neuron is enabled to learn any Boolean function. However, the neuron itself has no generalization capabilities. Thus, the simplest weightless neural network with such ability is known as *discriminator*. A discriminator is a single layer network with K RAM neurons capable of handling KN inputs. Therefore, a WiSARD network is composed of a set of these discriminators. Each one is responsible for classifying a different pattern. This way, the WiSARD network has the same number of inputs as its discriminators. Figure 8.1 illustrates the architecture of the WiSARD's discriminator.

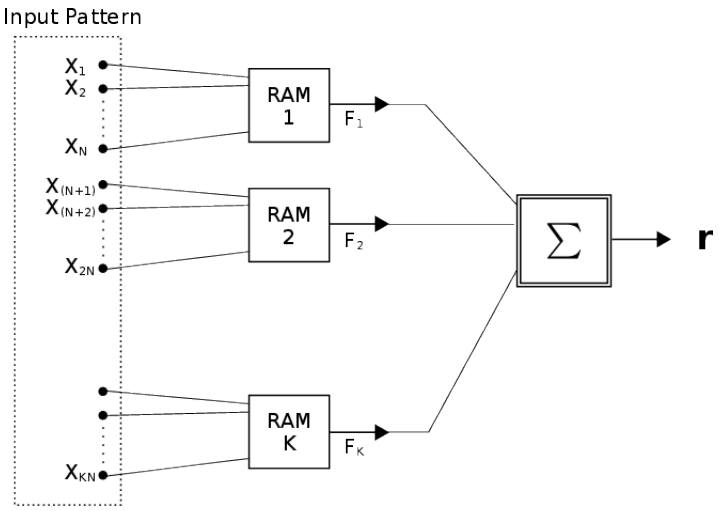


Figure 8.1: Discriminator: an elementary unity for the WiSARD model [9].

8.3 A Weightless Tracker

The proposed WiSARD tracker is composed of components called unit trackers (UT), which holds information about the bounding box of the object to be tracked and the WiSARD instance. In this tracker, the quantity of UTs is defined *a priori*. Each UT has also two search algorithms: a global one, which is responsible for searching over the entire image; and a local one, which searches around a local neighborhood.

In the beginning of the tracking task, the proposed solution requests all of its tracker units to perform a global search. After that, a local search is invoked, in order to improve the accuracy of the global search answer. If the resulting unit answer is greater than a minimal threshold, the location is stored in a history list.. When the second input image is presented, the tracker retrieves the last history entry, updating the bounding box position through local searches only.

As presented in section 8.2, the original RAM-based neuron of the WiSARD model stores a binary information about the presence or absence of a determined address (pattern). The RAM-based neural networks are subject to the overtraining problem. If the training set has many different patterns, most of RAMs composing the network may fill all (or almost all) available addresses. This event is called *saturation* and makes the network loose its classifying capabilities.

In order to overcome the overtraining problem, [97] proposed a WiSARD extension called DRASiW, which stores the RAM addressing frequency. This approach allows one to know which parts of the pattern (sub-patterns) happens more frequently. Furthermore, the remaining task is to isolate the relevant sub-patterns from the others [98]. A *bleaching* process is shown by [99], which proposes to accomplish this task by using the frequency information as a filter for the RAM-

neuron fire mechanism. When using this filter, a RAM is able to fire only when the frequency of the input address is greater or equal to a threshold, known as *bleaching threshold*.

During the tracking process, in addition the background and luminance disturbances, the moving objects tend to change their shape over time. In order to address this problem, two re-learning algorithms are proposed: *byMean*, which triggers the re-train procedure when the mean of the answers is less than a threshold, and *byDiff*, which accounts for the historical differences between answers in a buffer. The algorithm *byDiff* calls the re-learn procedure whenever the buffer reaches a given size and the sum of differences is greater than a threshold. Both algorithms increase the bleaching threshold before re-training.

Two local search algorithms have been developed as well: *linearLS* and *probLS*. The former explores the whole local neighborhood delimited by a number n of pixels around a given window position. The latter randomly chooses p different points (rounds) around the neighborhood also delimited by n pixels. Two global search algorithms have also been proposed. The first one, identified as *stepGS*, slides the unit window over the whole image, moving by n pixels at each step. Finally, the other one, identified as *threadedGS*, uses a grid of $n \times m$ instances of the *probLS* algorithm, associating each instance with a different thread.

8.4 Preliminary experimental setup and results

The proposed tracker has been evaluated using a part of the CAVIAR dataset[1] (the result of only one dataset is shown in this paper). Each image of this dataset has 384x288 pixels and before processing a frame, the tracker binarizes the input image using a luminance threshold. The tracker quality evaluation is done by calculating the amount of intersection between the bounding boxes given by the ground-truth data and the ones given by the tracker.

In order to evaluate the tracker, we assembled sixteen different combinations of the algorithms described in section 8.5, identified by *config0* to *config15*. The combinations of searching algorithms are divided into two pairs, as follows. The *searchPair1* has the global search *threadedGS* (16×16 *probLS*s, neighbourhood of 10 pixels, and 25 rounds), and the *linearLS* (neighbourhood of 10 pixels) as the local search algorithm. The *searchPair2* has the global search *stepGS* (step of 5 pixels), and the *probLS* (neighbourhood of 20 pixels and 40 rounds) as the local search algorithm.

Configs from 0 to 3 use the *searchPair1* and the *byDiff* (threshold equals to 0.08

[1]EC Funded CAVIAR project/IST 2001 37540, found at http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

and history size $n = 3$) re-learning strategy. Configs from 4 to 7 use the *searchPair2* and the *byDiff* (threshold equals to 0.08 and history size $n = 1$) re-learning strategy. Configs from 8 to 11 use *searchPair1* and the *byMean* (threshold equals to 0.08) re-learning algorithm, while configs from 9 to 15 use the *searchPair2* with the same re-relearning algorithm. The parameters used by the algorithms composing the configurations were found empirically.

We ran the tracker application, using the 16 aforementioned configurations, on an Intel(R) Core(TM) i7-3770 CPU 3.40GHz processor. We have used the UNIX time application to measure the processing time. Table 8.1 summarizes the results by showing the mean and standard deviation of each tracked target, as well as the quantity of Frames per Second for each configuration. Figure 8.2 shows an output image produced by the tracker.

Figure 8.2: An image from CAVIAR dataset and the bounding boxes drawn by the tracker.

Setup	Obj1		Obj2		Obj3		Obj4		FPS
	μ	σ	μ	σ	μ	σ	μ	σ	
config0	0.74	0.30	1.00	0.00	1.00	0.07	0.86	0.29	48
config1	0.70	0.21	0.98	0.10	0.76	0.17	1.00	0.04	56
config2	0.58	0.26	0.94	0.22	0.87	0.32	0.97	0.07	55
config3	0.87	0.20	0.95	0.20	0.69	0.25	0.98	0.05	52
config4	0.86	0.27	0.92	0.22	0.71	0.40	0.99	0.04	162
config5	0.68	0.19	0.94	0.18	1.00	0.01	0.94	0.14	166
config6	0.65	0.31	0.92	0.22	0.70	0.38	0.57	0.34	165
config7	0.74	0.29	0.88	0.23	0.82	0.36	0.79	0.26	158
config8	0.90	0.10	0.91	0.23	0.87	0.29	0.99	0.04	52
config9	0.64	0.24	1.00	0.00	0.52	0.31	0.97	0.10	56
config10	0.70	0.25	0.97	0.11	0.89	0.28	0.98	0.05	56
config11	0.62	0.26	0.94	0.20	0.88	0.28	0.94	0.14	51
config12	0.63	0.31	0.99	0.03	0.92	0.31	0.95	0.11	164
config13	0.68	0.25	0.93	0.19	0.70	0.42	0.97	0.08	170
config14	0.80	0.28	0.88	0.22	0.68	0.41	0.86	0.22	161
config15	0.86	0.20	0.58	0.22	0.80	0.33	0.82	0.26	153

Table 8.1: Summary of results.

The optimal values for the tracking quality measure are the mean equals to 1 and the standard deviation tending to 0, meaning the window is over the object all the time. Values lesser than 1 indicate the tracker has lost its target (or part of it) during the tracking time.

8.5 Hierarchical short and midterm memory

In order to improve the previous results, it was investigated a hierarchical memory-based tracker. Inspired by the human memory hierarchy, the proposed tracker is based on the concept of short- and medium-term memories. It is assumed that the shape changing of a object is seen as a new pattern to be learnt. For each pattern of the followed object, the proposed tracker stores a number of discriminators, each one representing a pattern learned in different moments of the tracking process. Thus, the hypothesis is that it is possible to keep tracking the object even if it changes its shape or becomes occluded for a period of time.

In the beginning of the process, the location of the object in the first frame is used as an input to the tracker, which trains the first discriminator and stores it in the hierarchic memory. For the next frames, the discriminator is used to find the object at the scene, locally searching around the last object's location. The discriminator returns a score to each position inside the searched region, and the position that returns the higher score is assumed to be the location of the object in the current frame. This process goes on until the classification score reaches a *pattern threshold*. When the score falls below this threshold, the tracker assumes that a new discriminator has to be trained in order to learn the new object shape. The tracker then proceeds to storing the current discriminator into the medium-term memory, and training a new discriminator to assume that position into the short-term memory.

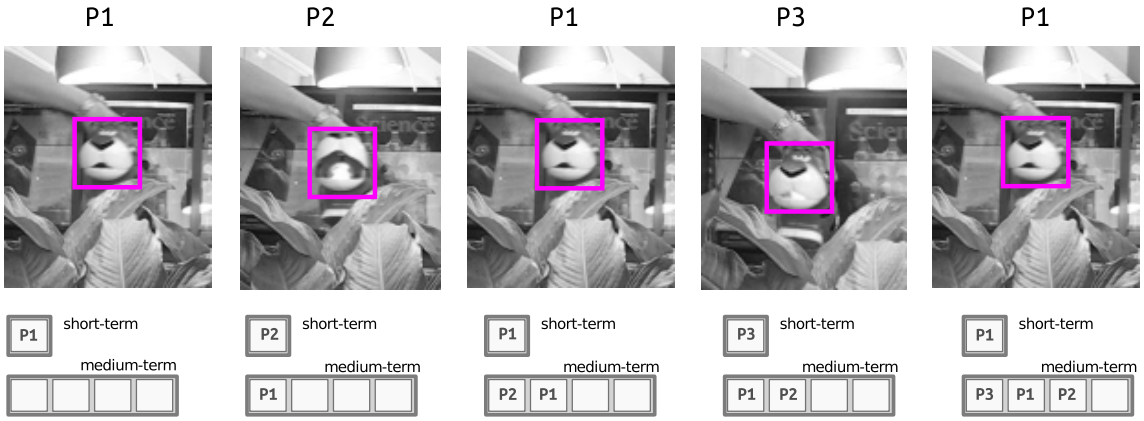


Figure 8.3: Hierarchic Memory example: At first, the discriminator P1 is used to find the object; in sequence, a new discriminator P2 is trained and placed in the first position; then, if the discriminator P1 returns the best score, it goes to the first place of the queue. In a future frame, P3 is trained and placed in the first position, then, if discriminator P1 returns the best score, it goes to the first position.

For each new frame, the tracker searches for the object pattern into both memory queues. The discriminator that gives the best score is chosen to represent the object location at the current frame, and that discriminator is transferred to the first position of the queue. Both queues have a maximum number of discriminators they can store. When this maximum number is reached, the discriminator located in the last position is dropped. This process guarantees that the most recently seen patterns are maintained in the hierarchic memory. Using this strategy, the discriminator that has not been used for the longest time, is naturally discarded when it is necessary to release memory to allocate a new discriminator. Figure 8.3 illustrates an example of allocation at the hierarchic memory with capacity to store four discriminators while Figure 8.4 shows some states of the tracker running in a real video clip.

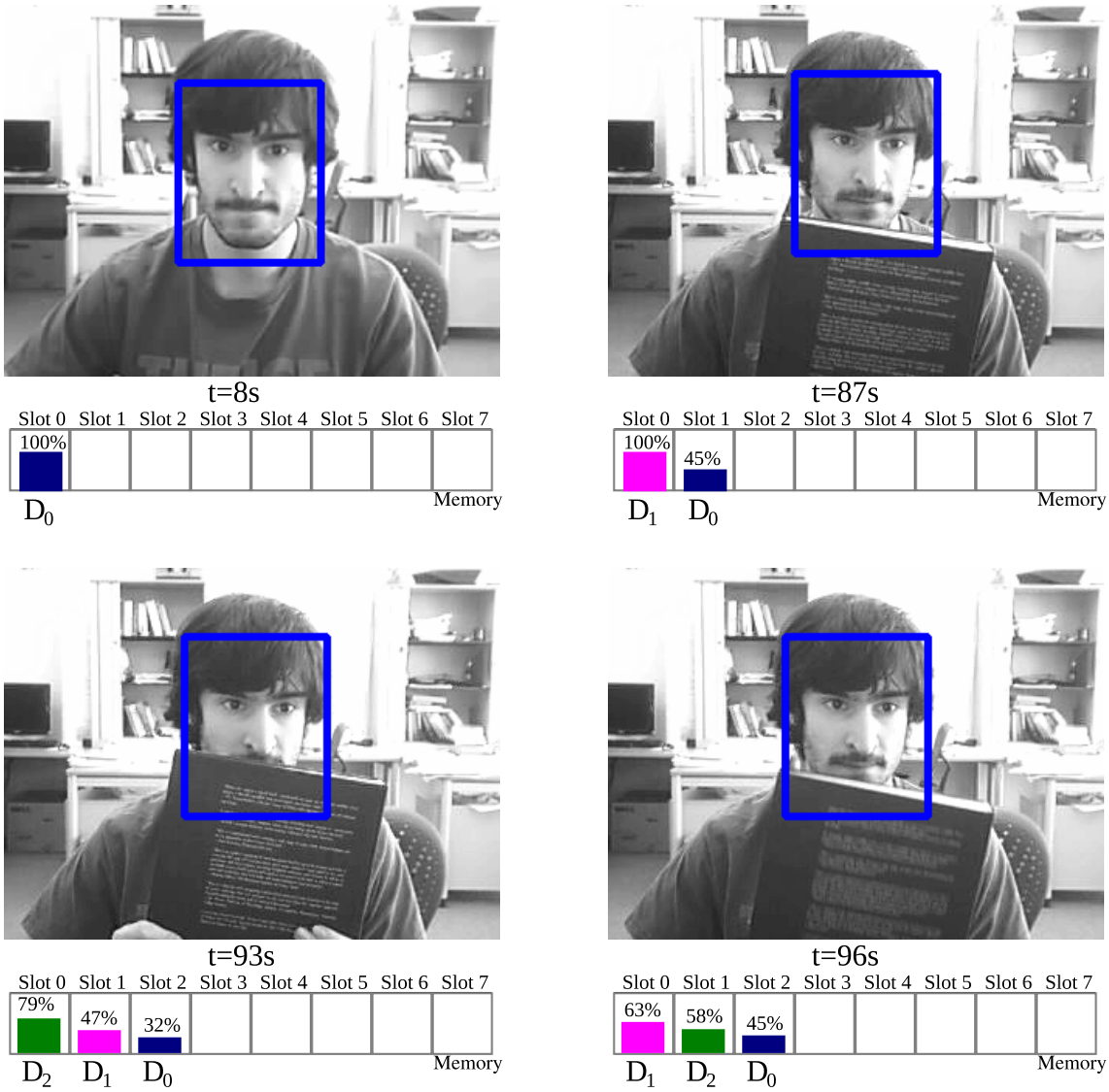


Figure 8.4: Running tracker in a video clip called *occluded face 2*. In the figure it is shown the ordering and creation of the discriminators in the memory queue.

8.5.1 Experimental setup and results

The *WHMTracker*[2] has been tested with default and tuned parameters in the same set of videos (Figure 8.5)[3] examined in [10]. The video clips names and the corresponding default and tuned parameters are shown in Table 8.2. All videos are in gray scale and present some problematic situations for a tracking system to handle, such as occlusion and shape changing over time. Before training a discriminator, the cropped image of the object, given by the bounding box, is binarized. For this purpose, the mean value of luminance is used as threshold. This process is employed while the tracker is searching for the object around a local neighborhood.

[2]The author would like to thank Daniel Nascimento for the tracker codification and experiments.

[3]The set of videos is available in: `http://vision.ucsd.edu/~bbabenko/project_miltrack.html`

The tracker is coded in C++ and we run in a 64bit machine running Linux kernel 2.6.

Figure 8.5: The set of videoclips as input for the considered tracker.

Table 8.2: Default and tuned parameters used in each tested video clip. Video clips identified with * indicate that a background extraction procedure is also part of the parameters.

Video	*Bits*	*New disc.*	*Memory Size*	*Search area*
Default params.	*5*	*0.7*	*6*	*12*
Tiger1*	default	0.35	20	14
Tiger2*	default	0.35	20	16
Occluded Face	3	0.5	10	10
Occluded Face 2	3	0.5	10	10
David Indoor	6	default	default	10
Sylvester	3	0.8	default	5

Table 8.3: Average Center Location Errors (in pixels). Values marked with '*' indicate the best performance and boldfaced ones represent the second best performances.

Video Clip	*MILTrack*	*WHMTrack*	*WHMTrackTuned*	*FPS*
Sylvester	**11**	22	8*	87
David Indoor	23	**11**	8*	22
Occluded Face	**27**	**27**	12*	17
Occluded Face 2	20	**16**	9*	28
Tiger 1	**16**	33	11*	45
Tiger 2	**18**	21	10*	43
Coupon Book	**15**	4*	4*	21

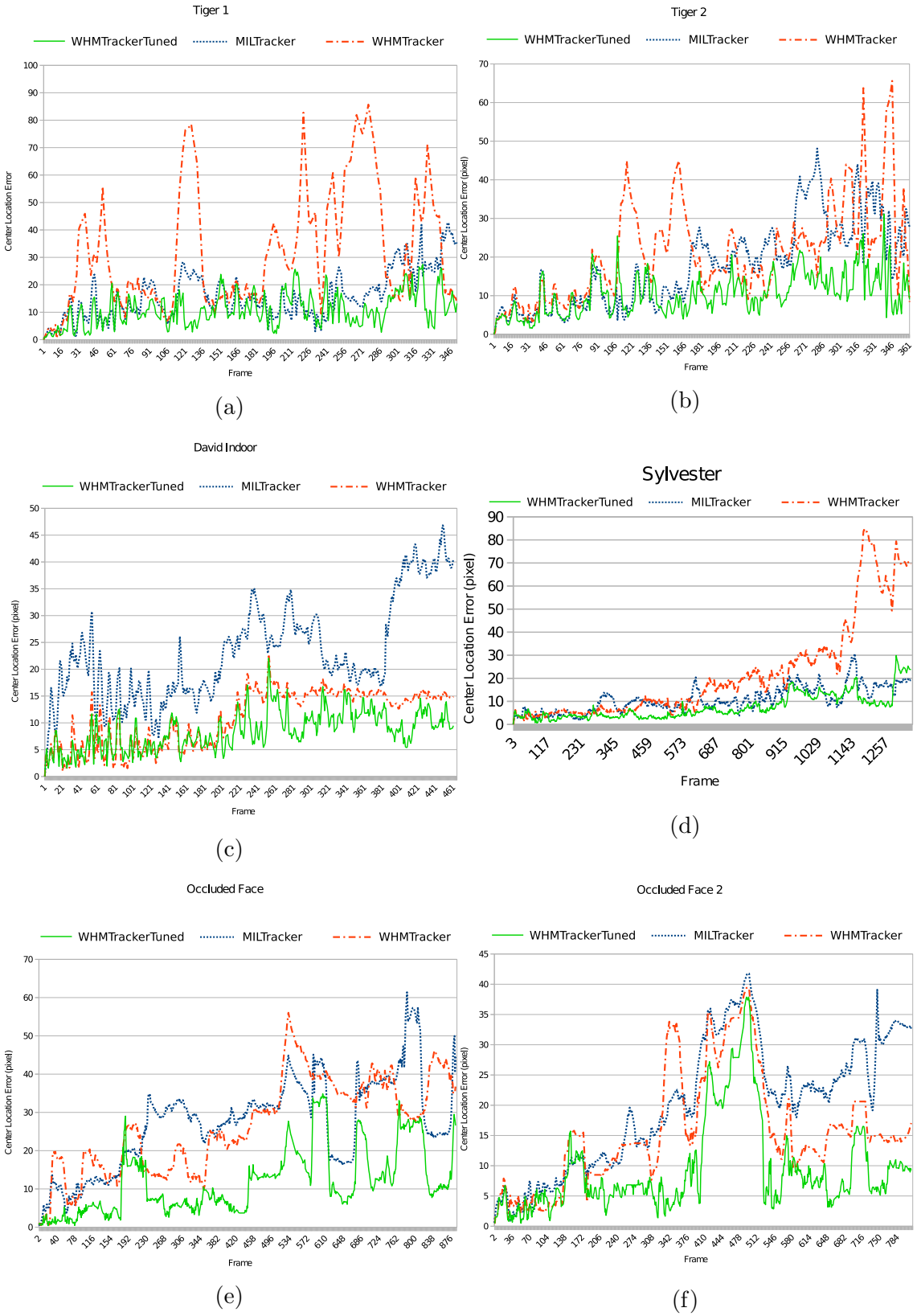


Figure 8.6: Figures 8.6a , 8.6b, 8.6c, 8.6d, 8.6e and 8.6f show a comparison among the tracker in [10], WHMTracker and Tuned WHMTracker for *Tiger1*, *Tiger2*, *DavidIndoor*, *Sylvester*, *OccludedFace* and *OccludedFace2* video clips, respectively.

In order to compare the results of the proposed tracker with the ones in [10], the average bounding box center error was adopted. Each video clip includes the associated ground truth data, which gives the position and size of the object from 5 to 5 frames. The same linear interpolation as in [10] was used to get the bounding box information for each frame. In addition, the tracker was executed 5 times for each video and the average error of the center location error was computed. Figure 8.6 shows the results for the set of video clips. Each plot has three pieces of information: the MILTrack result as well as the WHMTracker with and without tuned parameters. Table 8.3 shows a comparison between the results herein obtained and those in [10][4].

8.6 Final Remarks

In this chapter, the WiSARD model was evaluated as a solution in an on-line multi-object tracking application. Despite the use of a very limited input information (binarized image), the WiSARD model showed promising results towards an adequate classifying algorithm for this type of application. The further tracker, which uses a hierarchic memory architecture in order to store a queue of object patterns represented by discriminators of the WiSARD model was capable of getting satisfactory results. This memory architecture model was important to overcome problems such as occlusion, because a memory of a past seen object is stored and it is used as soon as the object becomes visible again.

As shown by the experiments, the proposed tracker is able to surpass the results presented in [10], using tuned parameters. The online training of a new discriminator representing a new object pattern was possible due to the WiSARD architecture, which allows for one shot learning. The main shortcoming of the proposed solution is the parameterization search. Future improvement includes search over the environment and object properties in order to propose a solution for auto tuning the tracker parameters.

[4]Some tracker demos are available in the companion website which is located at http://labia.cos.ufrj.br/publicacoes/artigos/weightless-hierarchy-memory-tracker

Chapter 9

General Conclusions

9.1 Achievements

This thesis considered the positioning of a set of agents under connectivity constraints. Some of the main measures of connectivity in graphs was revised as well as the main weighting functions to represent the behaviour of the connectivity signal, based on the distance between agents. Furthermore, the problem and the correlated approaches have been reviewed into Chapter 3. It could be seen that the problem is interesting even from the simple instance where two points should be connected. Moreover, when considering multiple endpoints for which the connection is required, the problem is even harder to handle.

The first development of this thesis consisted on modelling a scenario where a set of pursuers, subject to connectivity constraints, was designed to approximate a set of targets. Therefore, the problem has been modelled considering the Manhattan distance (and the vehicles moving into a grid), for which a quadratic objective function (minimize the quadratic distance among pursuers and targets) under linear constraints have been realized. In order to handle the scalability problem of solving instances with reasonable big sizes, it also has been studied a recurrent neural network method as a solver for problems such as this one. It could be seen that even for instances with multiple robots and targets and relays, the neural network solver has converged in less than 0.35 seconds, which could be useful for online instances.

The main issue of handling the problem as an optimization model is the dynamical structure of the network. For dealing with such problem, it has been considered the second smaller eigenvalue of the Laplacian of the graph, as a measure to be optimized. It has been taken an estimate of this number, since 0 could not be double root of the characteristic polynom of the Laplacian of the underlying graph, it has explored this property in order to propose a measure of connectivity. This measure is based on the calculation of the square sum of the determinant

of the Laplacian matrix, substituting some columns in a proper way. In order to obtain a model free of considering the network structure directly, it is used the gradient over this measure, guaranteeing that the considered agent will move under the restriction that this measures, when applied on the underlying graph composed by its neighbours, be over the minimum connectivity measure.

Furthermore, in order to fulfil the distributed requirement, it is proposed that the distribution of the model, in a way that a node j, for calculating its new position, only needs to know information about its 1-hop neighbourhood. Additionally, over the proposed measure, we defined the minimum connectivity measure, which could be calculated in a straightforward way needing only the neighbourhood information and the minimum value for considered the weighting function.

Beyond these approaches, some population-based heuristics have been investigated. This is useful for the calculation of the random global structures search. Some numerical experiments have been run using some test scenarios in order to show the feasibility of the proposed methods.

The other objective of this thesis consisted of the study and proposal of a generic object tracker in order to compose the vision of the drone. For this objective, the WiSARD weightless neural network has been considered as the classifier method for composing the tracker. This choice was useful for such online requirement because of the velocity of the training of the classifier. This feature allowed to propose a method with a re-learning stage. For this reason, once the object changed its shape, the classifier was able to retrain the new pattern. Furthermore, a mid- and short-term memory based tracker has also been proposed in order to store the past patterns with some expiration time. The considered tracker had been submitted on a benchmark for which was able to overcome the considered literature tracker.

9.2 Perspective for further research

In the connectivity problem, there are still some open opportunities to extend the research. As some future insights, the following had been considered:

1. **Handle the obstacle scenario**: in this branch there are two other considerations which are the moving obstacles and static scenarios. It is important to highlight the weighting function problem over such scenarios because it is difficult to simulate the behaviour of such signals because the material of the obstacles and their shape would impact directly on the signal strength. Maybe it could be harder to consider such weighting functions and the velocity of the computation should be a requirement even more strict.

2. **Consider the vehicles dynamic models**: in a way to become even more

realistic, maybe the control of the connectivity should also consider the vehicles dynamic model of movement.

3. **SER for dynamic graphs**: In the experiments with the use of the Schedule by Edge Reversal the oriented concurrency graph is built over the current network topology. This ensures the starvation- and deadlock-freedom properties of the algorithm. But after each sink has finished the calculation of its new position, a new proximity graph is generated to update the neighbourhood information. Therefore, maybe it can be interesting the investigation of the SER actuation over proximity graphs, and the presented test-bed could be a good start for considering an application scenario for such consideration.

4. **Consider the minimization static problem** In Chapter 3, we described some works dealing with the network design using a minimum number of vehicles. A straightforward result from this thesis could be to apply the solution to this problem, by changing the abandon target policy and the capture state implementation. Instead of abandon targets we add more relays, the network could stretch in order to provide a minimum number to stablish connectivity. In addition, instead of erasing targets from the system, the objective would become to find a zero value for the objective function. Thus, the number of employed relays would be an approximation to such problem.

5. **Hybrid models** Back on Chapter 5, we discussed about the number of possible unlabelled trees with n vertices. However, the number of possibilities may decrease over time, depending on the structure the network can become. The real number of considered trees should be accounted as the number of reachable trees. A reachable tree $T^{'}$ generated from a tree T is such that $dist(a \in T, a^{'} \in T^{'}) \leq \Delta$, where $dist(\cdot)$ is some distance metric and Δ is the maximum allowed displacement. If the number of reachable trees is a reasonable one, then a hybrid system can involve the recurrent neural network solver (as depicted in Chapter 4) or even considering the Quadratic Programming Model with Quadratic Solutions. A drawback of this could be the distributed version of it. Nevertheless, it could generate interesting results.

9.3 Publications and submissions

Published papers:

1. Rafael Lima de Carvalho, Lunlong Zhong, Felipe França, Félix Mora-Camino, *Dynamic Placement with Connectivity for RSNs based on a Primal-Dual*

Neural Network on Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013.

2. Rafael Lima de Carvalho, Danilo Carvalho, Priscila M. V. Lima, Félix Mora-Camino, Felipe M. G. França, *Online tracking of multiple objects using WiSARD* on Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2014.

3. Daniel Nascimento, Rafael Lima de Carvalho, Félix Mora-Camino, Priscila Lima, Felipe França, *A WiSARD-based multi-term memory framework for online tracking of objects* on Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2015.

Submitted papers:

1. Rafael Lima de Carvalho, Felipe França and Felix Mora-Camino *Collective intelligence of a swarm of pursuers and relays under connectivity constraints*, on IEEE Transactions on Robotics.

Bibliography

[1] SENEL, F., YOUNIS, M. "Optimized relay node placement for establishing connectivity in sensor networks". In: *Global Communications Conference (GLOBECOM), 2012 IEEE*, pp. 512–517, 2012. doi: 10.1109/GLOCOM.2012.6503164.

[2] OLSSON, P., KVARNSTROM, J., DOHERTY, P., et al. "Generating UAV communication networks for monitoring and surveillance". In: *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pp. 1070 –1077, dec. 2010. doi: 10.1109/ICARCV.2010.5707968.

[3] TEKDAS, O., KUMAR, Y., ISLER, V., et al. "Building a Communication Bridge With Mobile Hubs", *IEEE T. Automation Science and Engineering*, v. 9, n. 1, pp. 171–176, 2012.

[4] TEKDAS, O., PLONSKI, P. A., KARNAD, N., et al. "Maintaining connectivity in environments with obstacles." In: *ICRA*, pp. 1952–1957. IEEE, 2010.

[5] THUNBERG, J., ÖGREN, P. "A Mixed Integer Linear Programming approach to pursuit evasion problems with optional connectivity constraints", *Autonomous Robots*, v. 31, n. 4, pp. 333–343, 2011. ISSN: 0929-5593. doi: 10.1007/s10514-011-9247-y. Disponível em: <http://dx.doi.org/10.1007/s10514-011-9247-y>.

[6] ROOKER, M. N., BIRK, A. "Multi-robot exploration under the constraints of wireless networking", *Control Engineering Practice*, v. 15, n. 4, pp. 435 – 445, 2007. ISSN: 0967-0661. doi: http://dx.doi.org/10.1016/j.conengprac.2006.08.007.

[7] ZHANG, Y. "On the LVI-based Primal-Dual Neural Network for Solving Online Linear and Quadratic Programming Problems". In: *American Control Conference*, pp. 1351–1356, Porland, OR, USA, 2005.

[8] ALVES, D. S. F. *ReSATyrus: Geração Automatizada de Grafos de Compartilhamento*. Tese de Mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2014.

[9] ALEKSANDER, I., MORTON, H. *An introduction to Neural Computing*. Second edition ed. Berkshire House, London, UK, Thomson Computer Press, 1995.

[10] BABENKO, B., VARMA, N., DOLLÁR, P., et al. "Multiple Instance Learning with Manifold Bags". In: *International Conference on Machine Learning (ICML)*, Bellevue, WA, 2011.

[11] CLERC, M. "Standard Particle Swarm Optimisation". 2012. Disponível em: <https://hal.archives-ouvertes.fr/hal-00764996>. 15 pages.

[12] BLUM, C., MERKLE, D. *Swarm Intelligence: Introduction and Applications*. Natural Computing Series. Springer Berlin Heidelberg, 2008. ISBN: 9783540740896. Disponível em: <https://books.google.com.br/books?id=6Ky4bVPCXqMC>.

[13] EBERHART, R., KENNEDY, J. "A new optimizer using particle swarm theory". In: *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pp. 39 –43, oct 1995. doi: 10.1109/MHS.1995.494215.

[14] DORIGO, M. *Optimization, Learning and Natural Algorithms.* Tese de Doutorado, Politecnico di Milano, Italie, 1992.

[15] KARABOGA, D., BASTURK, B. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", *J. of Global Optimization*, v. 39, n. 3, pp. 459–471, nov 2007. ISSN: 0925-5001. doi: 10.1007/s10898-007-9149-x.

[16] KUSHLEYEV, A., MELLINGER, D., POWERS, C., et al. "Towards a Swarm of Agile Micro Quadrotors", *Auton. Robots*, v. 35, n. 4, pp. 287–300, nov. 2013. ISSN: 0929-5593. doi: 10.1007/s10514-013-9349-9. Disponível em: <http://dx.doi.org/10.1007/s10514-013-9349-9>.

[17] PUERTAS, L., ELSEN, J. "Earthquake in Peru Kills Hundreds". 2007. Disponível em: <http://www.nytimes.com/2007/08/16/world/americas/16cnd-peru.html?_r=0>.

[18] AZEVEDO, A. L. "Acidente em Mariana é o maior da História com barragens de rejeitos". 2015. Disponível em: <http://goo.gl/5ICpih>.

[19] SHEKHAR, S., FEINER, S. K., AREF, W. G. "Spatial Computing", *Commun. ACM*, v. 59, n. 1, pp. 72–81, dez. 2015. ISSN: 0001-0782. doi: 10.1145/2756547. Disponível em: <http://doi.acm.org/10.1145/2756547>.

[20] JOYNER, D., NGUYEN, M. V., PHILIPS, D. *Algorithmic Graph Theory and Sage*, v. 1. Version 0.8-r1991 ed. , GNU Free Documentation, May 2013.

[21] ESFAHANIAN, A.-H. "Connectivity Algorithms". In: Wilson, R., Beineke, L. (Eds.), *Structural Graph Theory*, pp. 268–281, Cambridge University Press, 2013.

[22] SCHRIJVER, A. "On the history of the transportation and maximum flow problems", *Mathematical Programming*, v. 91, n. 3, pp. 437–445, 2002. ISSN: 0025-5610. doi: 10.1007/s101070100259. Disponível em: <http://dx.doi.org/10.1007/s101070100259>.

[23] EVEN, S., TARJAN, R. "Network Flow and Testing Graph Connectivity", *SIAM Journal on Computing*, v. 4, n. 4, pp. 507–518, 1975. doi: 10.1137/0204043. Disponível em: <http://dx.doi.org/10.1137/0204043>.

[24] EVEN, S., EVEN, G. *Graph Algorithms*. 2nd ed. , Cambridge University Press, 2012.

[25] TARJAN, R. "Depth first search and linear graph algorithms", *SIAM Journal on Computing*, 1972.

[26] MATULA, D. "Determining edge connectivity in 0(nm)". In: *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pp. 249–251, Oct 1987. doi: 10.1109/SFCS.1987.19.

[27] HENZINGER, M., RAO, S., GABOW, H. "Computing vertex connectivity: new bounds from old techniques". In: *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pp. 462–471, Oct 1996. doi: 10.1109/SFCS.1996.548505.

[28] GODSIL, C., ROYLE, G. *Algebraic Graph Theory*, v. 207, *Graduate Texts in Mathematics*. volume 207 of Graduate Texts in Mathematics. Springer, 2001.

[29] ZAVLANOS, M., EGERSTEDT, M., PAPPAS, G. "Graph-theoretic connectivity control of mobile robot networks", *Proceedings of the IEEE*, v. 99, n. 9, pp. 1525–1540, Sept 2011. ISSN: 0018-9219. doi: 10.1109/JPROC.2011.2157884.

[30] MOHAR, B. "The Laplacian spectrum of graphs". In: *Graph Theory, Combinatorics, and Applications*, pp. 871–898. Wiley, 1991.

[31] MOHAR, B. "Laplace eigenvalues of graphs—a survey", *Discrete Mathematics*, v. 109, n. 1–3, pp. 171 – 183, 1992. ISSN: 0012-365X. doi: http://dx.doi.org/10.1016/0012-365X(92)90288-Q. Disponível em: <http://www.sciencedirect.com/science/article/pii/0012365X9290288Q>.

[32] HAGEN, L., KAHNG, A. "New spectral methods for ratio cut partitioning and clustering", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, v. 11, n. 9, pp. 1074–1085, Sep 1992. ISSN: 0278-0070. doi: 10.1109/43.159993.

[33] VON LUXBURG, U. "A tutorial on spectral clustering", *Statistics and Computing*, v. 17, n. 4, pp. 395–416, 2007. ISSN: 0960-3174. doi: 10.1007/s11222-007-9033-z. Disponível em: <http://dx.doi.org/10.1007/s11222-007-9033-z>.

[34] NASCIMENTO, M. C. V. *Metaheurísticas para o problema de agrupamento de dados em grafo*. Tese de Doutorado, ICMC-USP São Carlos, 2010.

[35] FIEDLER, M. "Algebraic connectivity of graphs", *Czechoslovak Mathematical Journal*, v. 23, n. 98, pp. 298–305, 1973.

[36] DE SOUZA ROCHA, I. *Vetor de Fiedler e as componentes de Perron*. Tese de Mestrado, Universidade Federal do Rio Grande do Sul, 2012.

[37] DE GENNARO, M., JADBABAIE, A. "Decentralized Control of Connectivity for Multi-Agent Systems". In: *Decision and Control, 2006 45th IEEE Conference on*, pp. 3628–3633, Dec 2006. doi: 10.1109/CDC.2006.377041.

[38] KIM, Y., MESBAHI, M. "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian", *Automatic Control, IEEE Transactions on*, v. 51, n. 1, pp. 116–120, Jan 2006. ISSN: 0018-9286. doi: 10.1109/TAC.2005.861710.

[39] DIXON, C., FREW, E. W. "Maintaining Optimal Communication Chains in Robotic Sensor Networks Using Mobility Control". In: *Proceedings of the 1st International Conference on Robot Communication and Coordination*, RoboComm '07, pp. 1:1–1:8, Piscataway, NJ, USA, 2007. IEEE Press. ISBN: 978-963-9799-08-0. Disponível em: <http://dl.acm.org/citation.cfm?id=1377868.1377870>.

[40] GIL, S., SCHWAGER, M., JULIAN, B. J., et al. "Optimizing communication in air-ground robot networks using decentralized control". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1964–1971, May 2010. doi: 10.1109/ROBOT.2010.5509622.

[41] GUPTA, P., KUMAR, P. "The capacity of wireless networks", *Information Theory, IEEE Transactions on*, v. 46, n. 2, pp. 388–404, Mar 2000. ISSN: 0018-9448. doi: 10.1109/18.825799.

[42] CHENG, X., DU, D.-Z., WANG, L., et al. "Relay sensor placement in wireless sensor networks", *Wirel. Netw.*, v. 14, n. 3, pp. 347–355, jun. 2008. ISSN: 1022-0038. doi: 10.1007/s11276-006-0724-8. Disponível em: <http://dx.doi.org/10.1007/s11276-006-0724-8>.

[43] LIN, G.-H., XUE, G. "Steiner tree problem with minimum number of Steiner points and bounded edge-length", *Information Processing Letters*, v. 69, n. 2, pp. 53 – 57, 1999. ISSN: 0020-0190. doi: 10.1016/S0020-0190(98)00201-4. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020019098002014>.

[44] CHEN, D., DU, D.-Z., HU, X.-D., et al. "Approximations for Steiner Trees with Minimum Number of Steiner Points", *J. of Global Optimization*, v. 18, n. 1, pp. 17–33, sep 2000. ISSN: 0925-5001. doi: 10.1023/A:1008384012064. Disponível em: <http://dx.doi.org/10.1023/A:1008384012064>.

[45] SENEL, F., YOUNIS, M. "Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation", *Comput. Commun.*, v. 34, n. 16, pp. 1932–1941, out. 2011. ISSN: 0140-3664. doi: 10.1016/j.comcom.2011.05.010. Disponível em: <http://dx.doi.org/10.1016/j.comcom.2011.05.010>.

[46] GAREY, M. R., GRAHAM, R. L., JOHNSON, D. S. "The Complexity of Computing Steiner Minimal Trees", *SIAM Journal on Applied Mathematics*, v. 32, n. 4, pp. 835–859, 1977. ISSN: 00361399. Disponível em: <http://www.jstor.org/stable/2100193>.

[47] BURDAKOV, O., DOHERTY, P., HOLMBERG, K., et al. "Relay Positioning for Unmanned Aerial Vehicle Surveillance", *I. J. Robotic Res.*, v. 29, n. 8, pp. 1069–1087, 2010. doi: 10.1177/0278364910369463. Disponível em: <http://dx.doi.org/10.1177/0278364910369463>.

[48] ZHANG, Y., QUILLING, M. "Optimal Backbone Generation for Robotic Relay Networks". In: *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pp. 1 –6, 31 2011-aug. 4 2011. doi: 10.1109/ICCCN.2011.6005922.

[49] LIAO, Z., ZHANG, S., CAO, J., et al. "Minimizing Movement for Target Coverage in Mobile Sensor Networks". In: *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pp. 194–200, 2012. doi: 10.1109/ICDCSW.2012.38.

[50] KUHN, H. W., YAW, B. "The Hungarian method for the assignment problem", *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[51] KEMPE, D., MCSHERRY, F. "A Decentralized Algorithm for Spectral Analysis". In: *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pp. 561–568, New York, NY, USA, 2004. ACM. ISBN: 1-58113-852-0. doi: 10.1145/1007352.1007438. Disponível em: <http://doi.acm.org/10.1145/1007352.1007438>.

[52] YAMAUCHI, B. "Frontier-based Exploration Using Multiple Robots". In: *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, pp. 47–53, New York, NY, USA, 1998. ACM. ISBN: 0-89791-983-1. doi: 10.1145/280765.280773. Disponível em: <http://doi.acm.org/10.1145/280765.280773>.

[53] LIN, Z., BROUCKE, M., FRANCIS, B. "Local control strategies for groups of mobile autonomous agents", *Automatic Control, IEEE Transactions on*, v. 49, n. 4, pp. 622–629, April 2004. ISSN: 0018-9286. doi: 10.1109/TAC.2004.825639.

[54] JI, M., EGERSTEDT, M. "Distributed Coordination Control of Multiagent Systems While Preserving Connectedness", *Robotics, IEEE Transactions on*, v. 23, n. 4, pp. 693–703, Aug 2007. ISSN: 1552-3098. doi: 10.1109/TRO.2007.900638.

[55] ZAVLANOS, M. M., PAPPAS, G. J. "Distributed Connectivity Control of Mobile Networks", *IEEE Transactions on Robotics*, v. 24, n. 6, pp. 1416–1428, 2008.

[56] ZAVLANOS, M., PAPPAS, G. "Distributed connectivity control of mobile networks". In: *Decision and Control, 2007 46th IEEE Conference on*, pp. 3591–3596, Dec 2007. doi: 10.1109/CDC.2007.4434525.

[57] *Design and use paradigms for Gazebo, an open-source multi-robot simulator*, v. 3, 2004. doi: 10.1109/iros.2004.1389727. Disponível em: <http://dx.doi.org/10.1109/iros.2004.1389727>.

[58] MICHAEL, N., ZAVLANOS, M., KUMAR, V., et al. "Maintaining Connectivity in Mobile Robot Networks". In: Khatib, O., Kumar, V., Pappas, G. (Eds.), *Experimental Robotics*, v. 54, *Springer Tracts in Advanced Robotics*, Springer Berlin Heidelberg, pp. 117–126, 2009. ISBN: 978-3-642-00195-6. doi: 10.1007/978-3-642-00196-3_14. Disponível em: <http://dx.doi.org/10.1007/978-3-642-00196-3_14>.

[59] ZHUANG, W., CHEN, X., TAN, J. "Maintaning Communication Links Using a Team of Mobile Robots", *Int. J. of Computers, Communications and Control*, v. VII, n. 1, pp. 184–195, March 2012.

[60] YANG, P., FREEMAN, R. A., GORDON, G. J., et al. "Decentralized estimation and control of graph connectivity for mobile sensor networks." *Automatica*, v. 46, n. 2, pp. 390–396, 2010. Disponível em: <http://dx.doi.org/10.1016/j.automatica.2009.11.012>.

[61] LINDHE, M., KEVICZKY, T., JOHANSSON, K. "Multi-robot path following with visual connectivity". In: *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, pp. 1466–1471, Nov 2011. doi: 10.1109/ACSSC.2011.6190261.

[62] COUCEIRO, M., ROCHA, R., FERREIRA, N. "A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms". In: *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pp. 327–332, Nov 2011. doi: 10.1109/SSRR.2011.6106751.

[63] TILLETT, J., RAO, T. M., SAHIN, F., et al. "Darwinian Particle Swarm Optimization". In: *2nd Indian International Conference on Artificial Intelligence*, pp. 1474–1487, 2005.

[64] DUAN, H., LUO, Q., SHI, Y., et al. "Hybrid Particle Swarm Optimization and Genetic Algorithm for Multi-UAV Formation Reconfiguration", *Computational Intelligence Magazine, IEEE*, v. 8, n. 3, pp. 16–27, Aug 2013. ISSN: 1556-603X. doi: 10.1109/MCI.2013.2264577.

[65] FURUKAWA, T., DURRANT-WHYTE, H., BOURGAULT, F., et al. "Time-optimal coordinated control of the relative formation of multiple vehicles".

In: *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, v. 1, pp. 259–264 vol.1, July 2003. doi: 10.1109/CIRA.2003.1222099.

[66] LOVÁSZ, L., PELIKÁN, J., VESZTERGOMBI, K. *Discrete Mathematics: Elementary and Beyond*. Discrete Mathematics: Elementary and Beyond. Springer, 2003. ISBN: 9780387955841. Disponível em: <https://books.google.com.br/books?id=zPlPXBSa__YC>.

[67] KIM, S., KOJIMA, M. "Exact Solutions of Some Nonconvex Quadratic Optimization Problems via SDP and SOCP Relaxations", *Comput. Optim. Appl.*, v. 26, n. 2, pp. 143–154, nov. 2003. ISSN: 0926-6003. doi: 10.1023/A:1025794313696. Disponível em: <http://dx.doi.org/10.1023/A:1025794313696>.

[68] TANK, D. W., HOPFIELD, J. J. "Neural Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", *IEEE Transactions on Circuits and Systems*, v. 5, n. 33, pp. 533–541, 1986.

[69] KAMEL, M. S., XIA., Y. "Cooperative recurrent modular neural networks for constrained optimization: a survey of models and applications", *Cognitive Neurodynamics*, v. 3, n. 1, pp. 47–81, 2009.

[70] XIA, Y., WANG, J. "Reurrent neural networks: design and applications". cap. Reccurent Neural Networks for Optimiztion: the State of the Art, CRC Press, 2001.

[71] LUENBERGER, D., YE, Y. *Linear and Nonlinear Programming*. International Series in Operations Research & Management Science. Springer US, 2008. ISBN: 9780387745022. Disponível em: <https://books.google.com.br/books?id=-pD62uvi9lgC>.

[72] AIGNER, M., ZIEGLER, G. "Cayley's formula for the number of trees". In: *Proofs from THE BOOK*, Springer Berlin Heidelberg, pp. 201–206, 2010. ISBN: 978-3-642-00855-9. doi: 10.1007/978-3-642-00856-6_30. Disponível em: <http://dx.doi.org/10.1007/978-3-642-00856-6_30>.

[73] BARBOSA, V. C. *An Introduction to Distributed Algorithms*. Cambridge, MA, USA, MIT Press, 1996. ISBN: 0-262-02412-8, 9780262024129.

[74] BARBOSA, V. C., GAFNI, E. "Concurrency in Heavily Loaded Neighborhood-constrained Systems", *ACM Trans. Program. Lang. Syst.*, v. 11, n. 4,

pp. 562–584, out. 1989. ISSN: 0164-0925. doi: 10.1145/69558.69560. Disponível em: <http://doi.acm.org/10.1145/69558.69560>.

[75] CALABRESE, A., FRANÇA, F. "A Randomised Distributed Primer for the Updating Control of Anonymous ANN s". In: Marinaro, M., Morasso, P. (Eds.), *ICANN '94*, Springer London, pp. 585–588, 1994. ISBN: 978-3-540-19887-1. doi: 10.1007/978-1-4471-2097-1_137. Disponível em: <http://dx.doi.org/10.1007/978-1-4471-2097-1_137>.

[76] JR., G. M. A., FRANÇA, F. M., MARTINHON, C. A. "Randomized generation of acyclic orientations upon anonymous distributed systems", *Journal of Parallel and Distributed Computing*, v. 69, n. 3, pp. 239 – 246, 2009. ISSN: 0743-7315. doi: http://dx.doi.org/10.1016/j.jpdc.2008.11.009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0743731508002025>.

[77] ALT, H. *Computational Discrete Mathematics*, v. 2122. 2001.

[78] RASHEDI, E., NEZAMABADI-POUR, H., SARYAZDI, S. "GSA: A Gravitational Search Algorithm", *Information Sciences*, v. 179, n. 13, pp. 2232 – 2248, 2009. ISSN: 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2009.03.004. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025509001200>. Special Section on High Order Fuzzy Sets.

[79] CLERC, M. *What is a Difficult Problem?* ISTE, 2010. ISBN: 9780470612163. doi: 10.1002/9780470612163. Disponível em: <http://dx.doi.org/10.1002/9780470612163>.

[80] A.E, E., J.E, S. *Introduction to Evolutionary Computing*. Springer, 2007.

[81] EIGEN, M., WINKLER, R. *Steps towards life: a perspective on evolution*. Stufen zum Leben. Oxford University Press, Incorporated, 1992. ISBN: 9780198547518. Disponível em: <https://books.google.com.br/books?id=R7QTAQAAIAAJ>.

[82] BANZHAF, W. *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*. The Morgan Kaufmann Series in Artificial Intelligence Series. Morgan Kaufmann Publishers, 1998. ISBN: 9781558605107. Disponível em: <https://books.google.com.br/books?id=1697qefFdtIC>.

[83] BROWNLEE, J. *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu.com, 2011. ISBN: 1446785068, 9781446785065.

[84] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st ed. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0201157675.

[85] CIVICIOGLU, P. "Backtracking Search Optimization Algorithm for numerical optimization problems", *Applied Mathematics and Computation*, v. 219, n. 15, pp. 8121 – 8144, 2013. ISSN: 0096-3003. doi: http://dx.doi.org/10.1016/j.amc.2013.02.017. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0096300313001380>.

[86] KENNEDY, J., EBERHART, R. "Particle swarm optimization". In: *Neural Networks, 1995. Proceedings., IEEE International Conference on*, v. 4, pp. 1942–1948 vol.4, Nov 1995. doi: 10.1109/ICNN.1995.488968.

[87] PETALAS, Y., PARSOPOULOS, K., VRAHATIS, M. "Memetic particle swarm optimization", *Annals of Operations Research*, v. 156, n. 1, pp. 99–127, 2007. ISSN: 0254-5330. doi: 10.1007/s10479-007-0224-y. Disponível em: <http://dx.doi.org/10.1007/s10479-007-0224-y>.

[88] ZAMBRANO-BIGIARINI, M., CLERC, M., ROJAS, R. "Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements". In: *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 2337–2344, June 2013. doi: 10.1109/CEC.2013.6557848.

[89] KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P., et al. "Optimization by simmulated annealing", *science*, v. 220, n. 4598, pp. 671–680, 1983.

[90] COELLO, C. A. C., VAN VELDHUIZEN, D. A., LAMONT, G. B. *Evolutionary algorithms for solving multi-objective problems*, v. 242. Springer, 2002.

[91] TALBI, E.-G. *Metaheuristics: from design to implementation*, v. 74. John Wiley & Sons, 2009.

[92] DRÉO, J. *Metaheuristics for hard optimization: methods and case studies*. Springer Science & Business Media, 2006.

[93] SABATTINI, L., SECCHI, C., CHOPRA, N., et al. "Distributed Control of Multirobot Systems With Global Connectivity Maintenance", *Robotics, IEEE Transactions on*, v. 29, n. 5, pp. 1326–1332, Oct 2013. ISSN: 1552-3098. doi: 10.1109/TRO.2013.2267971.

[94] SANMIGUEL, J., CAVALLARO, A., MARTÍNEZ, J. "Adaptive Online Performance Evaluation of Video Trackers", *IEEE Transactions on Image Processing*, v. 21, n. 5, pp. 2812 – 2823, 2012.

[95] PERNICI, F., BIMBO, A. D. "Object Tracking by Oversampling Local Features", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, v. in press, 2014. Disponível em: <http://www.micc.unifi.it/publications/2014/PD14>.

[96] FRANÇA, H., DA SILVA, J., DE GREGORIO, M., et al. "Movement persuit control of an offshore automated platform via a RAM-based neural network". In: *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pp. 2437 –2441, dec. 2010.

[97] DE GREGORIO, M. *On the reversibility of multi-discriminator systems*. Relatório Técnico Technical Report 125/97, Istituto di Cibernetica–CNR, 1997.

[98] CARVALHO, D. S., CARNEIRO, H. C. C., FRANÇA, F. M. G., et al. "B-bleaching: Agile Overtraining Avoidance in the WiSARD Weightless Neural Classifier", *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 515–520, abr. 2013.

[99] GRIECO, B. P., LIMA, P. M., GREGORIO, M. D., et al. "Producing pattern examples from "mental" images", *Neurocomputing*, v. 73, n. 7–9, pp. 1057 – 1064, 2010. ISSN: 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2009.11.015. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231210000159>.

Appendix A

Detailed results over the *allLeft* and *symmetric* scenarios

Table A.1: Summary of results for the scenario *allLeft* with behaviour scheme *allStatic*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.11|0.01	0.23|0.02	1.74|0.44	0.21|0.10	0.50|0.26	0.19|0.08	0.16|0.07	0.17|0.06	1.03|0.54	1.14|0.60	0.69|0.24	0.52|0.22	0.08|0.03	0.68|0.27
	OF	22.86	22.94	25.25	20.48	16.44	21.79	20.98	20.36	18.54	15.69	12.56	20.35	11.57	8.82
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7
	Simul. Sec	23	23	19	138	63	41	80	151	42	58	177	41	167	200
t_2	AvgProcTime	0.11|0.01	0.24|0.02	1.74|0.43	0.24|0.12	0.82|0.24	0.18|0.07	0.19|0.08	0.14|0.05	1.05|0.50	1.28|0.64	1.12|0.44	0.53|0.22	0.11|0.04	1.03|0.52
	OF	22.86	22.94	25.47	16.77	25.19	22.90	19.97	23.46	19.58	16.86	20.38	21.02	14.76	17.33
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	19	69	20	57	47	180	40	43	37	35	62	39
t_3	AvgProcTime	0.11|0.01	0.25|0.02	0.73|0.35	0.19|0.08	0.75|0.27	0.17|0.07	0.14|0.05	0.13|0.05	1.18|0.49	1.06|0.55	1.19|0.51	0.50|0.23	0.12|0.05	1.06|0.49
	OF	22.86	22.94	9.04	8.70	22.51	18.83	26.16	20.61	19.09	15.93	18.29	18.19	20.87	21.17
	Success	1.0	1.0	0.7	0.7	1.0	1.0	1.0	0.9	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	200	200	25	62	186	200	31	81	42	37	36	35
t_4	AvgProcTime	0.11|0.01	0.26|0.02	1.13|0.57	0.19|0.08	0.71|0.29	0.19|0.08	0.13|0.05	0.16|0.05	1.28|0.55	1.66|0.63	0.68|0.23	0.54|0.22	0.08|0.03	1.09|0.48
	OF	22.86	22.94	14.46	14.26	20.74	21.90	20.85	17.85	21.70	22.41	15.97	21.26	9.14	21.14
	Success	1.0	1.0	1.0	0.7	1.0	1.0	0.7	1.0	1.0	1.0	0.9	1.0	1.0	1.0
	Simul. Sec	23	23	46	200	28	41	200	110	27	26	200	35	183	35
t_5	AvgProcTime	0.11|0.01	0.28|0.02	1.64|0.54	0.29|0.14	0.38|0.17	0.17|0.07	0.16|0.06	0.18|0.07	1.25|0.48	1.38|0.66	0.60|0.28	0.57|0.24	0.08|0.03	1.07|0.52
	OF	22.86	22.94	22.90	17.11	11.42	20.57	21.36	18.27	22.56	21.23	18.45	21.83	8.74	20.21
	Success	1.0	1.0	1.0	1.0	0.6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	1.0
	Simul. Sec	23	23	23	40	200	93	90	65	27	39	67	30	200	37
t_6	AvgProcTime	0.11|0.01	0.29|0.02	1.42|0.59	0.29|0.14	0.37|0.17	0.19|0.08	0.20|0.08	0.17|0.07	1.30|0.49	1.68|0.60	0.44|0.23	0.46|0.18	0.08|0.03	1.24|0.54
	OF	22.86	22.94	20.99	17.55	11.55	21.57	20.98	20.37	22.23	22.66	21.32	17.65	14.84	23.16
	Success	1.0	1.0	1.0	1.0	0.7	1.0	1.0	1.0	1.0	1.0	0.9	1.0	0.9	1.0
	Simul. Sec	23	23	28	45	200	37	40	58	27	26	200	63	200	24
t_7	AvgProcTime	0.11|0.01	0.30|0.02	1.45|0.60	0.20|0.08	0.59|0.28	0.13|0.06	0.14|0.05	0.18|0.07	1.50|0.52	1.29|0.65	0.41|0.19	0.40|0.18	0.12|0.05	0.97|0.46
	OF	22.86	22.94	21.63	25.89	16.27	16.99	17.13	19.77	24.06	19.18	17.20	13.66	20.78	16.89
	Success	1.0	1.0	1.0	0.7	1.0	1.0	1.0	1.0	1.0	1.0	0.7	1.0	1.0	1.0
	Simul. Sec	23	23	26	200	41	165	171	64	22	45	200	100	40	42
t_8	AvgProcTime	0.11|0.01	0.31|0.03	1.57|0.55	0.18|0.08	0.73|0.27	0.16|0.07	0.15|0.06	0.15|0.05	1.08|0.56	1.57|0.58	0.42|0.19	0.46|0.20	0.10|0.04	1.00|0.47
	OF	22.86	22.94	22.91	9.27	21.65	23.54	18.45	15.72	19.59	22.58	16.52	17.77	18.38	17.86
	Success	1.0	1.0	1.0	0.7	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0	1.0

Table A.2: Summary of results for the scenario *allLeft* with behaviour scheme *allRandom*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.11|0.01	0.45|0.07	1.71|0.55	0.27|0.12	0.70|0.37	0.13|0.06	0.19|0.08	0.16|0.05	1.22|0.60	1.83|0.71	0.59|0.28	0.47|0.20	0.10|0.05	1.08|0.53
	OF	23.86	22.50	24.08	20.61	20.45	22.26	21.75	29.90	22.92	24.07	18.24	21.35	16.09	22.09
	Success	1.0	1.0	1.0	1.0	1.0	0.4	1.0	0.4	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	26	22	55	30	200	43	200	33	24	56	55	56	32
t_2	AvgProcTime	0.12|0.01	0.46|0.06	1.73|0.62	0.26|0.13	0.84|0.32	0.19|0.09	0.16|0.07	0.16|0.04	1.47|0.58	1.85|0.64	0.63|0.28	0.55|0.26	0.11|0.04	1.17|0.52
	OF	23.84	24.23	24.41	16.90	22.22	18.92	17.77	31.28	22.81	23.42	21.05	20.41	21.31	21.26
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.6	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	21	24	22	57	23	45	97	200	25	23	58	35	74	30
t_3	AvgProcTime	0.11|0.01	0.40|0.06	1.48|0.56	0.33|0.14	0.69|0.31	0.18|0.07	0.16|0.07	0.19|0.08	1.34|0.58	1.29|0.69	0.54|0.31	0.56|0.25	0.08|0.03	1.28|0.64
	OF	22.08	23.25	22.22	19.61	22.66	21.03	16.53	19.88	22.93	19.36	14.33	19.19	15.63	22.40
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0
	Simul. Sec	23	30	28	32	29	48	84	41	28	50	70	31	200	24
t_4	AvgProcTime	0.14|0.03	0.49|0.07	1.83|0.54	0.28|0.13	0.71|0.33	0.14|0.05	0.17|0.07	0.14|0.05	1.02|0.57	1.88|0.77	0.42|0.19	0.36|0.14	0.11|0.05	0.91|0.41
	OF	22.19	22.61	24.60	20.59	22.05	23.18	21.34	29.16	16.41	23.39	17.61	19.01	17.78	19.99
	Success	1.0	1.0	1.0	1.0	1.0	0.9	1.0	0.7	1.0	1.0	0.9	1.0	1.0	1.0
	Simul. Sec	28	23	22	48	30	200	53	200	46	25	200	196	55	61
t_5	AvgProcTime	0.11|0.01	0.50|0.06	1.64|0.66	0.27|0.13	0.77|0.34	0.13|0.05	0.14|0.05	0.14|0.05	1.36|0.65	1.74|0.79	0.44|0.19	0.47|0.23	0.12|0.05	1.25|0.60
	OF	24.84	22.63	22.72	17.62	23.28	21.59	19.12	26.36	22.06	22.77	17.09	17.39	21.56	22.78
	Success	1.0	1.0	1.0	1.0	1.0	0.9	1.0	0.6	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	19	27	26	58	24	200	176	200	29	26	196	57	35	26
t_6	AvgProcTime	0.12|0.01	0.55|0.15	1.79|0.62	0.19|0.08	0.56|0.29	0.19|0.08	0.17|0.07	0.14|0.05	1.14|0.56	2.06|0.76	0.43|0.20	0.58|0.23	0.10|0.05	1.35|0.50
	OF	22.78	23.92	25.13	19.28	20.21	16.46	17.88	24.88	19.17	22.68	16.52	22.04	15.36	24.58
	Success	1.0	1.0	1.0	0.7	1.0	1.0	1.0	0.9	1.0	1.0	0.9	1.0	1.0	1.0
	Simul. Sec	23	21	21	200	45	54	66	200	38	20	200	30	82	20
t_7	AvgProcTime	0.12|0.01	0.60|0.17	1.54|0.81	0.34|0.14	0.70|0.38	0.14|0.05	0.19|0.08	0.13|0.05	1.36|0.62	1.87|0.69	0.41|0.18	0.44|0.21	0.11|0.05	1.23|0.51
	OF	21.65	22.83	23.82	23.06	19.16	20.69	20.23	24.95	21.56	23.74	17.82	19.33	23.84	21.68
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0	0.9	1.0	1.0	1.0
	Simul. Sec	27	25	29	29	33	186	52	200	26	25	200	80	42	24
t_8	AvgProcTime	0.11|0.00	0.51|0.06	1.36|0.67	0.25|0.12	0.70|0.26	0.20|0.09	0.14|0.05	0.14|0.06	1.25|0.55	1.71|0.57	0.42|0.20	0.62|0.25	0.11|0.04	0.94|0.47
	OF	21.79	22.81	19.74	16.92	22.87	21.48	23.10	19.00	21.69	23.41	28.00	22.86	18.66	21.01
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0	1.0

Table A.3: Summary of results for the scenario *allLeft* with behaviour scheme *4Evasive4Static*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.11|0.01	0.64|0.09	1.55|0.55	0.19|0.08	0.71|0.30	0.16|0.06	0.24|0.13	0.18|0.07	1.23|0.50	1.74|0.81	0.42|0.20	0.52|0.23	0.08|0.03	1.17|0.50
	OF	22.33	22.22	22.70	18.17	23.72	17.61	19.97	19.89	22.40	23.13	12.89	20.19	19.83	19.28
	Success	1.0	1.0	1.0	0.7	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0	0.9	1.0
	Simul. Sec	24	24	23	200	25	77	47	49	29	26	200	34	200	35
t_2	AvgProcTime	0.11|0.01	0.65|0.09	1.66|0.47	0.25|0.13	0.53|0.26	0.13|0.05	0.13|0.06	0.18|0.07	1.36|0.49	1.53|0.62	0.76|0.30	0.46|0.19	0.08|0.03	0.75|0.26
	OF	22.33	22.22	23.76	16.49	20.00	32.62	16.59	17.68	22.93	24.05	22.05	18.31	14.29	9.24
	Success	1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.9
	Simul. Sec	24	24	21	59	51	200	173	54	25	27	31	51	200	200
t_3	AvgProcTime	0.11|0.01	0.66|0.10	0.72|0.34	0.24|0.12	0.79|0.27	0.15|0.06	0.13|0.05	0.17|0.08	1.19|0.48	1.18|0.62	0.44|0.25	0.55|0.24	0.11|0.04	1.07|0.47
	OF	22.33	22.22	15.58	12.58	24.88	17.50	20.68	17.62	20.12	18.14	17.60	18.73	16.85	18.79
	Success	1.0	1.0	0.7	1.0	1.0	1.0	0.9	1.0	1.0	1.0	0.9	1.0	1.0	1.0
	Simul. Sec	24	24	200	71	22	130	200	69	32	46	200	42	53	50
t_4	AvgProcTime	0.11|0.01	0.67|0.10	1.42|0.61	0.27|0.14	0.71|0.29	0.18|0.07	0.19|0.08	0.15|0.05	1.21|0.54	1.16|0.61	0.55|0.26	0.62|0.37	0.10|0.04	1.19|0.53
	OF	22.33	22.22	21.72	17.33	20.73	22.13	20.08	27.63	21.87	16.25	15.96	21.19	13.13	19.24
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.4	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	24	24	27	48	27	44	48	200	29	51	89	33	94	36
t_5	AvgProcTime	0.11|0.01	0.69|0.10	1.53|0.53	0.27|0.13	0.37|0.17	0.17|0.06	0.18|0.08	0.14|0.06	1.25|0.50	1.27|0.62	0.71|0.37	0.52|0.22	0.13|0.05	1.17|0.53
	OF	22.33	22.22	24.13	14.62	13.17	18.91	18.32	19.27	22.79	19.40	20.23	19.86	21.26	21.15
	Success	1.0	1.0	1.0	1.0	0.7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	24	24	23	51	200	90	56	127	26	40	40	41	33	30
t_6	AvgProcTime	0.11|0.01	0.70|0.10	1.57|0.52	0.22|0.11	0.36|0.17	0.18|0.07	0.19|0.08	0.17|0.07	0.62|0.31	1.07|0.58	0.44|0.23	0.48|0.25	0.08|0.03	0.99|0.50
	OF	22.33	22.22	22.69	16.89	10.66	19.17	20.42	21.89	11.73	13.73	18.49	15.92	24.38	16.44
	Success	1.0	1.0	1.0	1.0	0.7	1.0	1.0	1.0	0.7	1.0	0.9	1.0	0.9	1.0
	Simul. Sec	24	24	23	93	200	54	54	60	200	66	200	67	200	50
t_7	AvgProcTime	0.11|0.01	0.71|0.10	1.52|0.55	0.24|0.12	0.74|0.28	0.17|0.06	0.14|0.05	0.19|0.08	1.44|0.50	1.74|0.60	0.73|0.39	0.56|0.24	0.08|0.03	1.13|0.49
	OF	22.33	22.22	21.60	15.35	24.64	20.32	22.19	19.25	24.04	24.03	18.75	19.60	19.67	19.14
	Success	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	1.0
	Simul. Sec	24	24	24	67	24	62	200	45	22	23	39	38	200	34
t_8	AvgProcTime	0.11|0.01	0.72|0.10	1.72|0.46	0.19|0.08	0.64|0.29	0.17|0.07	0.15|0.06	0.19|0.08	1.49|0.44	1.23|0.61	0.41|0.25	0.60|0.32	0.09|0.04	0.79|0.37
	OF	22.33	22.22	25.15	10.31	21.32	17.92	19.31	18.72	23.92	18.37	12.37	19.53	11.98	16.51
	Success	1.0	1.0	1.0	0.6	1.0	1.0	1.0	1.0	1.0	1.0	0.7	1.0	1.0	0.9

Table A.4: Summary of results for the scenario *allLeft* with behaviour scheme *4Collaborative4Spiral*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.14|0.03	1.16|0.43	1.70|0.45	0.30|0.15	0.89|0.25	0.20|0.09	0.20|0.10	0.21|0.09	1.62|0.42	1.77|0.60	0.70|0.36	0.66|0.27	0.12|0.05	1.18|0.56
	OF	21.48	21.48	24.70	19.61	24.32	22.35	20.98	20.98	24.96	24.04	19.72	22.70	22.90	22.09
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	17	34	17	29	30	26	16	18	30	20	29	21
t_2	AvgProcTime	0.14|0.03	1.17|0.43	1.79|0.42	0.42|0.13	0.71|0.27	0.14|0.06	0.20|0.09	0.21|0.11	1.66|0.42	2.07|0.52	0.79|0.32	0.62|0.25	0.13|0.06	1.38|0.61
	OF	21.48	21.48	24.88	24.36	21.34	22.70	21.12	21.19	24.98	24.54	21.86	22.24	20.27	24.43
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	16	18	24	101	28	30	16	16	25	20	28	17
t_3	AvgProcTime	0.14|0.03	1.19|0.44	1.86|0.40	0.38|0.13	0.92|0.21	0.20|0.09	0.20|0.10	0.20|0.09	1.50|0.47	1.71|0.58	0.90|0.31	0.70|0.25	0.13|0.06	1.24|0.55
	OF	21.48	21.48	25.68	23.45	25.55	21.47	19.15	23.60	24.97	22.77	23.13	25.00	19.98	23.41
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	15	19	16	31	31	29	18	20	20	16	28	19
t_4	AvgProcTime	0.14|0.03	1.21|0.45	1.87|0.46	0.33|0.15	0.94|0.21	0.20|0.10	0.20|0.09	0.20|0.10	1.54|0.48	1.94|0.53	0.95|0.25	0.71|0.26	0.17|0.06	1.25|0.53
	OF	21.48	21.48	25.63	21.80	25.82	22.46	20.58	19.30	24.27	24.29	24.12	23.65	23.92	22.74
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	15	29	15	30	28	30	18	17	18	18	17	21
t_5	AvgProcTime	0.14|0.03	1.23|0.46	1.93|0.35	0.33|0.14	0.89|0.21	0.23|0.10	0.21|0.10	0.19|0.09	1.53|0.49	1.75|0.64	0.88|0.29	0.65|0.26	0.13|0.06	1.14|0.55
	OF	21.48	21.48	26.15	23.43	25.56	23.36	21.15	20.55	24.73	22.31	22.90	23.87	21.71	21.87
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	15	28	16	20	28	28	17	20	21	19	25	22
t_6	AvgProcTime	0.14|0.03	1.24|0.46	1.77|0.44	0.42|0.14	0.92|0.23	0.21|0.10	0.21|0.10	0.20|0.09	1.50|0.47	1.93|0.49	0.78|0.36	0.67|0.26	0.12|0.05	1.25|0.53
	OF	21.48	21.48	25.34	23.22	24.40	20.84	21.55	21.91	24.59	24.35	20.42	23.88	20.86	23.54
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	16	18	16	26	25	28	17	17	26	18	31	19
t_7	AvgProcTime	0.14|0.03	1.27|0.47	1.79|0.47	0.41|0.13	0.85|0.26	0.19|0.09	0.19|0.09	0.17|0.07	1.49|0.45	1.86|0.56	0.73|0.33	0.64|0.27	0.12|0.05	1.41|0.49
	OF	21.48	21.48	24.68	24.23	24.58	18.58	22.06	20.79	23.89	24.82	20.27	22.78	21.85	24.54
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Simul. Sec	23	23	16	18	17	35	29	58	17	17	29	19	29	17
t_8	AvgProcTime	0.14|0.03	1.28|0.48	1.86|0.41	0.42|0.12	0.82|0.27	0.21|0.09	0.16|0.08	0.18|0.09	1.64|0.42	1.81|0.56	0.93|0.29	0.73|0.26	0.12|0.06	1.31|0.54
	OF	21.48	21.48	25.76	24.24	24.36	20.70	17.64	18.66	24.88	23.76	23.85	25.07	21.01	23.26
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table A.5: Summary of results for the scenario *symmetric* with behaviour scheme *allStatic*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.07|0.03	0.44|0.21	0.71|0.25	0.18|0.06	0.35|0.12	0.16|0.04	0.16|0.04	0.15|0.04	0.75|0.23	0.74|0.30	0.36|0.15	0.35|0.11	0.08|0.03	0.72|0.23
	OF	34.10	35.14	24.00	23.87	22.19	29.32	34.05	26.62	44.26	20.96	19.12	30.86	33.22	25.35
	Success	1.0	1.0	0.0	0.0	0.0	0.4	0.2	0.1	0.5	0.1	0.0	0.4	0.4	0.2
	Simul. Sec	199	143	200	200	200	200	200	200	200	200	200	200	200	200
t_2	AvgProcTime	0.07|0.03	0.44|0.21	0.71|0.25	0.18|0.06	0.36|0.12	0.15|0.04	0.16|0.04	0.15|0.04	0.73|0.26	0.85|0.33	0.36|0.15	0.37|0.11	0.08|0.03	0.73|0.24
	OF	34.10	35.14	24.33	21.54	24.26	25.55	29.25	31.18	41.31	29.14	19.82	32.05	31.13	23.09
	Success	1.0	1.0	0.0	0.0	0.0	0.1	0.4	0.2	0.5	0.4	0.0	0.4	0.4	0.4
	Simul. Sec	199	143	200	200	200	200	200	200	200	200	200	200	200	200
t_3	AvgProcTime	0.07|0.03	0.45|0.22	0.71|0.25	0.18|0.06	0.36|0.13	0.16|0.04	0.15|0.04	0.15|0.04	0.69|0.23	0.73|0.30	0.37|0.15	0.36|0.11	0.08|0.03	0.73|0.23
	OF	34.10	35.14	18.65	24.89	23.51	31.63	31.58	29.63	28.31	24.65	25.54	35.62	24.60	32.59
	Success	1.0	1.0	0.0	0.0	0.0	0.2	0.2	0.2	0.4	0.1	0.1	0.4	0.1	0.4
	Simul. Sec	199	143	200	200	200	200	200	200	200	200	200	200	200	200
t_4	AvgProcTime	0.07|0.03	0.46|0.22	0.71|0.25	0.18|0.06	0.36|0.13	0.16|0.04	0.16|0.05	0.16|0.04	0.71|0.25	0.81|0.31	0.39|0.15	0.39|0.12	0.08|0.03	0.74|0.24
	OF	34.10	35.14	23.87	24.33	19.19	27.51	28.80	29.56	31.41	22.90	25.40	27.69	19.49	25.78
	Success	1.0	1.0	0.0	0.0	0.0	0.1	0.4	0.2	0.5	0.4	0.2	0.4	0.2	0.2
	Simul. Sec	199	143	200	200	200	200	200	200	200	200	200	200	200	200
t_5	AvgProcTime	0.07|0.03	0.46|0.22	0.71|0.24	0.18|0.06	0.36|0.12	0.16|0.04	0.16|0.04	0.15|0.04	0.68|0.23	0.81|0.30	0.39|0.15	0.36|0.12	0.08|0.03	0.72|0.24
	OF	34.10	35.14	24.86	25.56	20.73	29.47	29.42	22.20	28.75	30.32	19.72	23.16	27.68	24.12
	Success	1.0	1.0	0.0	0.0	0.0	0.2	0.2	0.1	0.4	0.4	0.2	0.2	0.4	0.2
	Simul. Sec	199	143	200	200	200	200	200	200	200	200	200	200	200	200
t_6	AvgProcTime	0.07|0.03	0.47|0.22	0.70|0.24	0.18|0.06	0.36|0.13	0.15|0.04	0.16|0.05	0.16|0.04	0.64|0.25	0.78|0.30	0.39|0.15	0.37|0.11	0.08|0.03	0.71|0.23
	OF	34.10	35.14	22.80	23.79	20.92	27.49	25.28	31.28	26.52	20.03	25.89	24.78	31.57	33.10
	Success	1.0	1.0	0.0	0.0	0.0	0.1	0.2	0.1	0.2	0.4	0.2	0.4	0.4	0.4
	Simul. Sec	199	143	200	200	200	200	200	200	200	200	200	200	200	200
t_7	AvgProcTime	0.07|0.03	0.47|0.23	0.71|0.24	0.18|0.06	0.35|0.13	0.16|0.04	0.16|0.04	0.15|0.05	0.68|0.24	0.80|0.29	0.39|0.15	0.37|0.12	0.08|0.03	0.73|0.23
	OF	34.10	35.14	18.67	21.41	23.89	30.84	29.17	26.96	44.77	29.38	24.28	33.55	22.90	32.76
	Success	1.0	1.0	0.0	0.0	0.0	0.2	0.1	0.1	0.5	0.2	0.2	0.4	0.1	0.4
	Simul. Sec	199	143	200	200	200	200	200	200	200	200	200	200	200	200
t_8	AvgProcTime	0.07|0.03	0.48|0.23	0.71|0.25	0.18|0.06	0.36|0.12	0.16|0.04	0.16|0.04	0.16|0.04	0.61|0.24	0.82|0.31	0.36|0.14	0.37|0.11	0.09|0.03	0.76|0.24
	OF	34.10	35.14	23.98	20.35	24.40	28.52	31.60	28.28	23.11	30.56	20.37	28.22	33.78	31.87
	Success	1.0	1.0	0.0	0.0	0.0	0.2	0.4	0.4	0.1	0.5	0.0	0.2	0.4	0.4

Table A.6: Summary of results for the scenario *symmetric* with behaviour scheme *allRandom*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.07|0.04	0.32|0.16	0.73|0.25	0.18|0.06	0.36|0.12	0.15|0.04	0.16|0.04	0.15|0.04	0.70|0.23	0.83|0.29	0.39|0.15	0.38|0.11	0.08|0.03	0.77|0.21
	OF	35.09	27.13	27.35	28.52	28.01	35.89	34.72	38.35	20.50	35.44	23.11	28.14	35.54	28.72
	Success	1.0	0.2	0.0	0.0	0.0	0.4	0.2	0.4	0.1	0.2	0.1	0.2	0.1	0.2
	Simul. Sec	183	200	200	200	200	200	200	200	200	200	200	200	200	200
t_2	AvgProcTime	0.05|0.01	0.41|0.16	0.73|0.25	0.18|0.06	0.38|0.12	0.15|0.04	0.16|0.05	0.16|0.04	0.67|0.25	0.80|0.28	0.37|0.15	0.38|0.11	0.09|0.02	0.76|0.22
	OF	25.29	37.76	24.74	25.12	24.68	31.84	29.86	33.23	22.87	28.73	31.71	31.36	26.27	24.76
	Success	0.4	0.9	0.0	0.0	0.1	0.1	0.2	0.4	0.1	0.2	0.0	0.2	0.4	0.4
	Simul. Sec	200	200	200	200	200	200	200	200	200	200	200	200	200	200
t_3	AvgProcTime	0.06|0.02	0.72|0.71	0.83|0.29	0.19|0.06	0.37|0.13	0.16|0.04	0.16|0.04	0.16|0.04	0.76|0.22	0.84|0.28	0.39|0.14	0.39|0.10	0.09|0.03	0.76|0.22
	OF	36.63	40.27	24.25	26.39	25.61	29.39	31.47	32.41	33.27	29.38	24.71	29.42	23.63	26.20
	Success	0.5	1.0	0.8	0.0	0.0	0.4	0.4	0.2	0.5	0.2	0.1	0.4	0.1	0.2
	Simul. Sec	200	130	200	200	200	200	200	200	200	200	200	200	200	200
t_4	AvgProcTime	0.05|0.01	0.34|0.17	0.74|0.25	0.19|0.06	0.36|0.13	0.15|0.04	0.16|0.04	0.16|0.04	0.62|0.23	0.79|0.28	0.40|0.14	0.38|0.11	0.08|0.03	0.78|0.22
	OF	29.41	28.57	31.18	25.72	34.00	31.49	26.35	34.60	20.90	30.74	24.37	24.91	31.47	28.91
	Success	0.4	0.2	0.0	0.1	0.0	0.2	0.5	0.4	0.0	0.1	0.2	0.1	0.1	0.2
	Simul. Sec	200	200	200	200	200	200	200	200	200	200	200	200	200	200
t_5	AvgProcTime	0.05|0.01	0.47|0.18	0.76|0.25	0.19|0.06	0.37|0.12	0.16|0.04	0.16|0.04	0.17|0.04	0.71|0.22	0.81|0.29	0.40|0.15	0.37|0.11	0.09|0.02	0.78|0.22
	OF	29.71	26.99	24.79	21.47	23.92	27.02	34.47	33.44	22.92	22.76	24.63	26.95	32.51	38.95
	Success	0.9	0.4	0.1	0.0	0.0	0.2	0.2	0.2	0.1	0.2	0.1	0.1	0.4	0.4
	Simul. Sec	200	200	200	200	200	200	200	200	200	200	200	200	200	200
t_6	AvgProcTime	0.06|0.03	0.69|0.72	0.74|0.25	0.18|0.06	0.37|0.13	0.16|0.04	0.13|0.04	0.16|0.04	0.69|0.23	0.80|0.28	0.37|0.14	0.37|0.11	0.09|0.02	0.73|0.22
	OF	37.20	38.87	29.46	30.66	20.85	34.85	26.11	24.01	27.50	19.71	29.56	27.15	35.41	29.76
	Success	1.0	1.0	0.1	0.0	0.1	0.4	0.0	0.6	0.1	0.1	0.0	0.2	0.2	0.1
	Simul. Sec	182	151	200	200	200	200	200	200	200	200	200	200	200	200
t_7	AvgProcTime	0.06|0.02	0.41|0.18	0.73|0.25	0.19|0.06	0.36|0.12	0.16|0.04	0.16|0.04	0.16|0.04	0.75|0.22	0.82|0.29	0.41|0.14	0.40|0.10	0.09|0.02	0.77|0.23
	OF	36.89	31.40	19.08	20.05	21.71	34.00	29.99	28.48	30.69	34.04	33.15	31.55	36.49	33.75
	Success	1.0	0.4	0.0	0.1	0.0	0.2	0.4	0.1	0.2	0.1	0.4	0.5	0.4	0.2
	Simul. Sec	150	200	200	200	200	200	200	200	200	200	200	200	200	200
t_8	AvgProcTime	0.05|0.01	0.42|0.17	0.72|0.25	0.20|0.06	0.37|0.12	0.15|0.04	0.16|0.04	0.16|0.04	0.76|0.21	0.76|0.30	0.36|0.14	0.39|0.11	0.09|0.02	0.78|0.22
	OF	33.31	31.91	19.09	24.95	32.59	31.68	32.93	29.54	34.35	25.23	30.26	34.26	32.47	27.12
	Success	0.4	0.4	0.0	0.2	0.0	0.2	0.2	0.5	0.2	0.1	0.0	0.4	0.4	0.4

Table A.7: Summary of results for the scenario *symmetric* with behaviour scheme *4Evasive4Static*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.07|0.04	0.66|0.35	0.71|0.25	0.18|0.07	0.36|0.12	0.16|0.04	0.16|0.04	0.15|0.04	0.71|0.25	0.71|0.29	0.38|0.15	0.37|0.12	0.08|0.03	0.68|0.23
	OF	36.47	35.15	23.30	22.38	23.18	30.47	27.04	28.07	28.93	20.16	21.50	30.43	26.28	11.16
	Success	1.0	1.0	0.0	0.0	0.0	0.4	0.2	0.2	0.4	0.0	0.1	0.4	0.4	0.0
	Simul. Sec	143	158	200	200	200	200	200	200	200	200	200	200	200	200
t_2	AvgProcTime	0.07|0.04	0.67|0.36	0.70|0.26	0.18|0.06	0.36|0.13	0.18|0.08	0.15|0.04	0.14|0.05	0.73|0.24	0.81|0.31	0.36|0.14	0.36|0.12	0.09|0.03	0.74|0.24
	OF	36.47	35.15	26.57	21.39	21.68	21.80	25.06	25.66	33.40	28.73	18.78	22.35	30.92	33.80
	Success	1.0	1.0	0.0	0.0	0.0	0.1	0.1	0.1	0.5	0.4	0.0	0.2	0.4	0.4
	Simul. Sec	143	158	200	200	200	200	200	200	200	200	200	200	200	200
t_3	AvgProcTime	0.07|0.04	0.68|0.36	0.71|0.25	0.18|0.06	0.35|0.12	0.17|0.05	0.13|0.04	0.15|0.04	0.71|0.25	0.75|0.30	0.38|0.16	0.37|0.12	0.08|0.03	0.73|0.24
	OF	36.47	35.15	24.38	26.45	24.47	27.33	20.04	25.55	40.45	19.02	21.07	32.42	18.70	25.22
	Success	1.0	1.0	0.0	0.0	0.0	0.1	0.0	0.2	0.5	0.1	0.1	0.4	0.1	0.2
	Simul. Sec	143	158	200	200	200	200	200	200	200	200	200	200	200	200
t_4	AvgProcTime	0.07|0.04	0.68|0.36	0.71|0.25	0.18|0.06	0.36|0.12	0.17|0.04	0.16|0.04	0.15|0.04	0.73|0.26	0.77|0.31	0.36|0.14	0.38|0.12	0.08|0.03	0.73|0.24
	OF	36.47	35.15	22.51	20.13	21.43	25.11	28.69	26.51	27.97	24.84	16.20	32.79	23.91	36.43
	Success	1.0	1.0	0.0	0.0	0.0	0.1	0.2	0.2	0.5	0.4	0.0	0.4	0.4	0.4
	Simul. Sec	143	158	200	200	200	200	200	200	200	200	200	200	200	200
t_5	AvgProcTime	0.07|0.04	0.69|0.37	0.71|0.25	0.18|0.06	0.36|0.13	0.16|0.06	0.16|0.04	0.16|0.04	0.71|0.25	0.84|0.32	0.38|0.15	0.38|0.12	0.08|0.03	0.69|0.23
	OF	36.47	35.15	25.06	25.49	20.71	29.79	26.94	31.51	31.44	28.30	25.12	27.39	34.42	19.07
	Success	1.0	1.0	0.0	0.0	0.0	0.0	0.2	0.2	0.4	0.4	0.1	0.4	0.4	0.1
	Simul. Sec	143	158	200	200	200	200	200	200	200	200	200	200	200	200
t_6	AvgProcTime	0.07|0.04	0.69|0.37	0.70|0.25	0.18|0.06	0.36|0.13	0.17|0.05	0.16|0.04	0.16|0.04	0.68|0.25	0.83|0.31	0.38|0.15	0.37|0.12	0.08|0.03	0.75|0.24
	OF	36.47	35.15	26.22	23.87	24.42	26.12	28.93	33.28	34.14	30.68	22.34	22.85	25.13	30.78
	Success	1.0	1.0	0.0	0.0	0.0	0.2	0.2	0.2	0.5	0.4	0.1	0.4	0.1	0.4
	Simul. Sec	143	158	200	200	200	200	200	200	200	200	200	200	200	200
t_7	AvgProcTime	0.07|0.04	0.70|0.37	0.71|0.25	0.18|0.06	0.35|0.12	0.16|0.04	0.15|0.04	0.16|0.04	0.70|0.24	0.86|0.31	0.38|0.15	0.37|0.13	0.08|0.03	0.75|0.24
	OF	36.47	35.15	22.97	28.42	24.58	29.26	26.76	34.43	45.05	30.30	16.66	28.89	18.61	31.06
	Success	1.0	1.0	0.0	0.0	0.0	0.2	0.1	0.4	0.5	0.5	0.1	0.4	0.1	0.4
	Simul. Sec	143	158	200	200	200	200	200	200	200	200	200	200	200	200
t_8	AvgProcTime	0.07|0.04	0.70|0.37	0.71|0.26	0.18|0.06	0.36|0.13	0.15|0.04	0.15|0.04	0.15|0.04	0.67|0.25	0.86|0.30	0.40|0.15	0.34|0.12	0.08|0.03	0.69|0.23
	OF	36.47	35.15	20.61	23.95	19.59	26.77	29.49	27.82	34.36	34.75	30.90	19.71	26.59	18.11
	Success	1.0	1.0	0.0	0.0	0.0	0.1	0.1	0.1	0.4	0.5	0.4	0.1	0.1	0.1

Table A.8: Summary of results for the scenario *symmetric* with behaviour scheme *4Collaborative4Spiral*.

		*SwarmLambda*2		*BSA*			*GA*			*SA*			*PSO*2011		
		Normal	*SER*	1	2	3	1	2	3	1	2	3	1	2	3
t_1	AvgProcTime	0.06|0.02	0.55|0.33	0.81|0.30	0.20|0.07	0.40|0.14	0.15|0.05	0.15|0.05	0.14|0.04	0.71|0.26	0.81|0.31	0.39|0.15	0.36|0.11	0.08|0.03	0.77|0.25
	OF	25.13	25.51	28.92	32.12	29.66	34.60	33.98	37.08	33.44	31.25	40.28	40.28	47.32	31.95
	Success	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0	0.9	0.8	0.9	1.0
	Simul. Sec	71	70	134	193	154	158	170	193	200	168	200	200	200	147
t_2	AvgProcTime	0.06|0.02	0.56|0.33	0.78|0.26	0.20|0.07	0.41|0.14	0.15|0.04	0.15|0.05	0.16|0.05	0.68|0.24	0.80|0.28	0.40|0.15	0.38|0.13	0.09|0.02	0.81|0.26
	OF	25.13	25.51	36.15	31.16	31.45	37.68	31.46	28.45	38.62	35.34	29.48	30.06	36.36	43.18
	Success	1.0	1.0	0.9	1.0	1.0	0.8	1.0	1.0	0.9	0.8	1.0	1.0	0.8	0.9
	Simul. Sec	71	70	200	160	161	200	166	112	200	200	175	133	200	200
t_3	AvgProcTime	0.06|0.02	0.56|0.33	0.84|0.29	0.20|0.06	0.39|0.13	0.14|0.04	0.15|0.05	0.14|0.04	0.71|0.26	0.82|0.32	0.41|0.16	0.38|0.14	0.08|0.03	0.73|0.22
	OF	25.13	25.51	26.66	31.33	33.00	37.05	32.49	42.44	31.78	31.74	31.38	29.97	42.01	39.13
	Success	1.0	1.0	1.0	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	1.0	0.9	0.9
	Simul. Sec	71	70	142	200	200	200	130	200	200	200	200	133	200	200
t_4	AvgProcTime	0.06|0.02	0.56|0.33	0.78|0.26	0.20|0.07	0.39|0.13	0.15|0.05	0.15|0.05	0.15|0.05	0.70|0.24	0.85|0.32	0.40|0.16	0.36|0.11	0.08|0.03	0.73|0.23
	OF	25.13	25.51	32.66	31.06	35.36	34.42	26.02	30.98	31.39	30.79	30.84	34.54	42.60	39.30
	Success	1.0	1.0	1.0	1.0	0.8	1.0	1.0	1.0	0.9	1.0	1.0	0.8	0.9	0.9
	Simul. Sec	71	70	195	154	200	188	119	132	200	150	154	200	200	200
t_5	AvgProcTime	0.06|0.02	0.57|0.33	0.79|0.26	0.20|0.07	0.39|0.13	0.14|0.04	0.15|0.05	0.15|0.05	0.70|0.23	0.81|0.29	0.39|0.14	0.37|0.11	0.08|0.03	0.71|0.23
	OF	25.13	25.51	33.70	30.20	30.60	35.88	28.01	34.90	33.74	39.80	39.30	38.79	43.89	40.79
	Success	1.0	1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.8	0.9	0.9	0.8	0.9	0.9
	Simul. Sec	71	70	200	162	171	199	143	147	200	200	200	200	200	200
t_6	AvgProcTime	0.06|0.02	0.57|0.34	0.83|0.28	0.20|0.07	0.40|0.14	0.15|0.04	0.15|0.05	0.15|0.04	0.73|0.25	0.77|0.30	0.41|0.16	0.36|0.11	0.08|0.02	0.73|0.23
	OF	25.13	25.51	29.79	32.89	33.24	44.89	34.09	41.03	31.67	33.33	35.30	43.90	40.00	35.02
	Success	1.0	1.0	1.0	1.0	1.0	0.9	1.0	0.9	1.0	0.9	0.9	0.8	0.9	0.9
	Simul. Sec	71	70	155	167	189	200	163	200	150	200	200	200	200	200
t_7	AvgProcTime	0.06|0.02	0.58|0.34	0.79|0.26	0.19|0.06	0.39|0.13	0.15|0.04	0.16|0.07	0.14|0.04	0.71|0.28	0.81|0.29	0.40|0.15	0.36|0.12	0.09|0.03	0.73|0.23
	OF	25.13	25.51	28.91	29.07	34.50	37.42	40.42	36.01	33.11	36.02	37.51	43.30	42.07	31.45
	Success	1.0	1.0	0.5	0.9	0.8	0.9	1.0	0.9	1.0	0.9	0.8	0.9	0.9	0.9
	Simul. Sec	71	70	200	200	200	200	189	200	129	200	200	200	200	200
t_8	AvgProcTime	0.06|0.02	0.58|0.34	0.78|0.29	0.20|0.06	0.40|0.15	0.16|0.05	0.15|0.04	0.15|0.04	0.71|0.23	0.83|0.31	0.39|0.14	0.38|0.13	0.08|0.03	0.72|0.23
	OF	25.13	25.51	32.41	33.81	28.51	31.79	36.80	39.33	36.25	31.70	37.83	29.63	42.04	34.43
	Success	1.0	1.0	1.0	0.9	1.0	1.0	0.9	1.0	0.9	1.0	0.9	1.0	0.9	0.9