הטכניון
מכון טכנולוגי לישראל
TECHNION
Israel Institute of Technology

ספריות הטכניון
The Technion Libraries

בית הספר ללימודי מוסמכים ע"ש ארווין וג'ואן ג'ייקובס
Irwin and Joan Jacobs Graduate School

Deployment Strategies for Coverage Control Problems

Yoav Palti

Deployment Strategies for Coverage Control Problems

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering

Yoav Palti

Submitted to the Senate
of the Technion — Israel Institute of Technology
TAMUZ 5779 Haifa July 2019

This research was carried out under the supervision of Prof. Daniel Zelazo, in the Faculty of Aerospace Engineering.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author's masters research period, the most up-to-date versions of which being:

Yoav Palti and Daniel Zelazo. A Projected Lloyd ' s Algorithm for Coverage Control Problems. In *Proceedings of The 59th Israel Annual Conference on Aerospace Sciences*, pages 1008–1022, Tel Aviv & Haifa, march 2019.

Acknowledgements

I would like to thank my advisor, Associate Professor Daniel Zelazo, for his great support during the work of this thesis, for letting me express myself and my ideas. I would like to thank MAFAT for partial support of this work.

Contents

List of Figures

Abstract

This work considers the problem of *coverage control* for large areas, where the number of sensors is limited and can achieve only partial coverage. To solve this problem, we are taking advantage of moving sensors - the sensors can move to achieve full coverage, under the limitation that on a given time only part of the area is covered. In the literature we find two main *coverage strategies* for doing so – trajectory based and partitioned based. In the trajectory-based approach, we are building a trajectory for each agent, such that the total area covered by all the agents is the total area need to be covered. A second strategy consist of partitioning (or tiling) the area into partitions that we can *deploy* our agents, and "jumping" between the partitions provides full coverage.

In this work, we consider the partial coverage problem with another constraint that requires the sensors to maintain partial coverage of some sub area at all time. This can represent, for example, maintaining a communication connection to a home base. Due to this constraint, we chose to focus on a partitioning strategy rather than a trajectory-based approach. We introduce an algorithm that provides a partitioning of an area while maintaining an intersection with some sub-area - the *projected Lloyd's Algorithm* (PLA). The PLA is a variation of the well-known Lloyd's Algorithm for partitioning and ensuring the coverage of some sub-area.

Furthermore, there might be a requirement for coverage control while maintaining some sort of formation. A classic example is a formation with the goal of locating an emitter, applying some sort of triangulation algorithm. Therefore, we propose a controller which combines the coverage controller (whether the PLA or Lloyd's algorithm based) with a distance-based formation controller, while maintaining a formation. We then analyze the quality of the combination. Each controller presented in this work is backed up with simulations and numerical results.

Abbreviations and Notations

$\mathbb{R}$	:	Set of real numbers
$\mathbb{N}$	:	Set of natural numbers
$\mathcal{G}$	:	A graph
$\mathcal{E}$	:	Set of graph edges
e_i	:	The i'th edge of a graph (vector)
e_{ij}	:	An edge connecting nodes i, j
w_{ij}	:	Vertices i, j relative position.
δ	:	A formation error
δ_k	:	The k'th edge error on a formation
$\mathcal{V}$	:	Set of graph nodes (vertices)
$R(x)$	:	Rigidity matrix
$E(\mathcal{G})$	:	Incidence matrix of graph $\mathcal{G}$
$V(x)$	:	Lyapunov function
PLA	:	Projected Lloyd's Algorithm.
MIR	:	Minimally Infinitesimally Rigid (graph).

Chapter 1

Introduction

Monitoring some range using autonomous agents is a well known problem that is widely researched. Common examples includes drones filming an area, robotic vacuum cleaners cleaning a room, satellites trying to geo-locate an emitter, or UAV's trying to locate a target [1, 2, 3]. A common problem with monitoring some range is when the sensors are limited in their sensing capability. For example, let us assume that we only have three drones with a limited resolution camera, and the mission is monitoring a farm from intruding animals. Let us further also assume that the farm is large enough such that the drones can't cover the whole area using one static formation. For achieving full coverage, the drones must be dynamic, and use some tactic to cover the whole farm. The situation is illustrated on Figure 1.1, where drones are covering some part of the farm. Another possible constraint can be some sub-range which must be at least partially covered at all times. For example, we want to make sure that the farm's pen is monitored, so we can recognize when a wolf is breaking in. Partial coverage in this case should be fine - the operator can recognize when the sheep are coming out of their peace to identify some possible intruder. The situation is illustrated on Figure 1.1, where the coverage area is intersecting with the sheep pen.

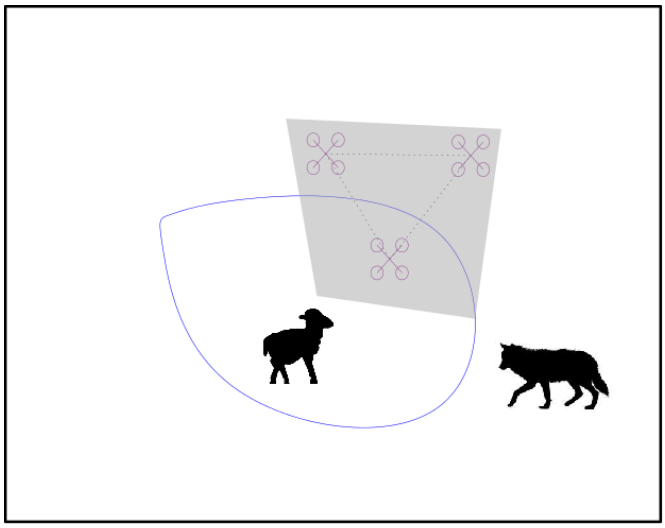


Figure 1.1: An example of an area needed to be covered (limited within the black rectangle) with subarea constraint (the blue drop-shape). The grey area represents the actual area covered.

Another interesting question is what happens when a spatial formation of those agents must be maintained. For example, the agents are performing geo-location of free animals in the farm. As formation control is a difficult optimization problem on its own, performing formation control under coverage constraints might lead to a conflict in the optimization process (coverage vs. formation), thus making this problem non-trivial. The problems explained above are two issues that this work tries to tackle.

In the next subsection, a discussion of the latest work in this field will be given, followed by this thesis contribution and organization.

1.1 Literature Review

In the recent decades, researchers have come to the conclusion that performing tasks using multi-agents networks can be beneficial [4]. An example for this application is monitoring some range, which is known in the scientific community as the *coverage control* problem [5].

Sensor coverage can be generally described as the reflection of how well a given range is monitored by sensors. This field is well explored in the scientific community, especially in the recent years as the fields of distributed algorithms and cooperative systems were developed. A concept that repeats in many works is finding a set of trajectories (at least one route for every sensor), allowing the mobile sensors to achieve the required coverage goal. For example, when trying to cover an area using two mobile sensors, the algorithm will define a trajectory for each sensor such that the whole area will be covered [6, 7, 8]. Another idea that appears in most of the relevant literature is the use of Voronoi diagrams for optimizing the sensor location [9, 10]. Those methods generally require prior knowledge on the area to be covered.

A possible approach is the study of static (or stationary) sensing networks [11]. This approach is very useful given we have constant access to the area needed to be covered, and that we have enough sensors. When the mission involves exploring, it is obvious that the sensors should move. Static networks allows us to determine some of their properties, such as sensing capabilities, more easily than dynamic ones. For example, in [11] the problem of sensing grid reliability is discussed. A network is reliable if it's connected (i.e. each sensor can communicate with another sensor, directly or indirectly) and covered (every point within the range needed to be covered is indeed covered).

When coverage is required for large areas, and specifically when a single sensor cannot provide full coverage of that area, the idea of using a *deployment* of mobile sensors is presented [5, 9]. In [12], the problem is formed using a probabilistic network model and some density function, to model detection of events. Then, an optimization algorithm is proposed to find the optimal deployment such that maximum coverage is obtained with minimal communications cost. In [9], Cortes *et al.* propose an optimization algorithm for maximal coverage using Centroidal Voronoi Tessellations (CVT) [13]. To achieve this partitioning, they are utilizing a continuous gradient controller implementation of the *Lloyd's algorithm* [14], an algorithm that employs a simple iterative method to compute the CVT. The former concept will be utilized and modified in this thesis to achieve new coverage capabilities that will be further described.

However, there are some cases where maintaining spatial properties are also as important as keeping the required range covered. One example is when the communications between the sensors are limited to some range. Another example is when the spatial shape is critical for the mission, as in geolocation missions, where we might need to perform some kind of triangulation. There are several solutions for this problem, known generally as the *formation control* problem [15, 16].

In the literature, we do find interesting examples of combining formation control and coverage control. For example, in [17] the authors tried to maximize the coverage in an environment cluttered with obstacles, and proposed an optimal dynamic formation control with a leader-following class formation, where the constraint is between the agents and the leader. In [18], the authors propose a receding horizon algorithm combined with Voronoi-based coverage control. In [19], the authors proposed a "mixing function", which defines how to mix sensors reading, resulting with interesting probabilistic properties (for example, to minimize the variance of a sensor reading). To the best of our knowledge, a combination between distance-based formation control and Voroni-based coverage control was never made.

1.2 Thesis Contribution

In this work, we formulate the coverage problem from another angle. We firstly assume that the given range can be at most *partially* covered. However, we also define some subrange that must be at least partially covered at all times (for example, surveillance or

home-base communications constraint). We present an algorithm for partitioning this range into tiles, such that each tile intersects with this sub-range constraint, and then, utilizing the approach outlined in [9] to achieve coverage of this tile. To achieve this intersection, we present a new, modified version of Lloyd's algorithm - the *projected Lloyd's algorithm*. This new algorithm modifies the original Lloyd's algorithm and utilizes a specific projection function to achieve partitioning answering the sub-range constraint. With this method, we can guarantee complete coverage of the range by sequentially deploying the agents from each tile to tile while ensuring that these tiles all have a non-empty intersection with the sub-range of interest.

We then take into account spatial constraints. We present a method to combine a deployment algorithm with a distance based formation controller. While the formation controller tries to maintain a required formation, the controller is trying to optimize the deployment of the sensors (maximal coverage), thus creating new dynamics where both the formation and coverage are not optimal. We propose to combine the controllers using a coefficient that balances between the controllers - in some of the cases, the deployment will have more weight, and on others - the formation will have more weight. We will demonstrate the influence of this coefficient and discuss its benefits and disadvantages.

1.3 Thesis Organization

The organization of this work is as follows. Chapter 2 covers the mathematical tools related to this work. It follows by the formal problem formulation in Chapter 3. In Chapter 4 the main solution to the problem is shown, and some simulations are given. Expanding the deployment strategy for formation control combined with deployment is given in Chapter 5. Finally, in Chapter 6 we conclude this work.

Chapter 2

Preliminaries

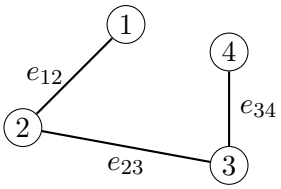
This chapter deals with the basic mathematical concepts and topics which are used in this thesis.

2.1 Graph Theory

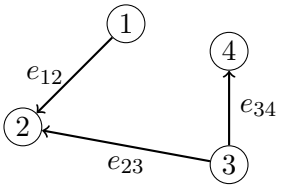
A Graph is a mathematical object that represents a set of elements which are related in some sense [20]. For example, a common use is a set of agents as the "elements", and the relations can be their communications [4]. As it appears that graphs have properties and implications on physical elements or systems, graph analysis can become very useful in the study of multi-agent systems. *Graph theory* is the mathematical study of graphs and the main tool for such an analysis.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair of two finite sets - the *vertex* (nodes) set $\mathcal{V}$ ($|\mathcal{V}| = n$), and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each vertex can represent a mathematical or physical entity, for example, a sensor. Each edge represents some relation between two entities, for example, communication connectivity or congruent sensing ranges among mobile sensors. An edge connecting nodes $v_i, v_j \in \mathcal{V}$ is denoted $e_{ij} = \{v_i, v_j\}$. If an orientation is assigned to an edge such that $e_{ij} = (v_i, v_j)$ is described by an ordered pair, then the graph is called *directed*. An *undirected graph* (see Figure 2.1) is one where if $e_{ij} = (v_i, v_j) \in \mathcal{E}$, then $e_{ji} = (v_j, v_i)$ is also an edge in $\mathcal{E}$. A *directed graph* is a graph where $e_{ij} = (v_i, v_j) \in \mathcal{E}$ does not imply $e_{ji} = (v_j, v_i) \in \mathcal{E}$, and the edge e_{ij} direction is from its *tail* i to its *head* j. A vertex degree (for an undirected graph) is the number of edges connected to it, and marked for vertex i as d_i. A *path* in a graph is a sequence of edges and vertices connecting two vertices. If there exists a path between every two vertices, then the graph is called a *connected* graph. A *subgraph* is a graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, such that $\mathcal{V}' \subset \mathcal{V}, \mathcal{E}' \subset \mathcal{E}$. A *cycle* is a connected graph where for each node, $d_i = 2$. A *tree* is a connected graph that contains no cycle subgraph. The *spanning tree* of a graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ that is a tree.

Graphs can be represented as matrices. The mathematical theory behind this is the *algebraic graph theory* [20]. Analyzing those matrices can sometimes simplify studying



(a) Undirected connected graph.



(b) A connected graph with arbitrary orientation assigned to edges.

Figure 2.1: An example of connected graph which is also a tree. Therefore, removing any edge will make this graph not connected.

the graph properties, and moreover - can sometimes help us represent a dynamical system using a graph [4].

One useful matrix is the *incidence matrix* of a directed graph, which not only describes connectivity between edges, but also indicates the direction of it. It is possible to define an incidence matrix of an undirected graph simply by assigning arbitrary directions to the edges (this also means that for undirected graph, the incidence matrix is not unique).

Definition 2.1.1. The *incidence matrix* of an undirected graph $\mathcal{G} = (V, \mathcal{E})$ with an arbitrary orientation, is the matrix $E(\mathcal{G})$ such that

$$[E(\mathcal{G})]_{ij} = \begin{cases} 1, & v_i \in \mathcal{V} \text{ is head of edge } e_j \in \mathcal{E}, \\ -1, & v_i \in \mathcal{V} \text{ is tail of edge } e_j \in \mathcal{E}, \\ 0, & otherwise. \end{cases}$$

In Figure 2.1(b), we can see a graph with arbitrary edge orientation, and its incidence matrix is:

$$E(\mathcal{G}) = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 1 \end{bmatrix}.$$

2.2 Rigidity Theory

Rigidity theory is a mathematical tool that describes what happens to a structure under some perturbation (such as pushing it), usually being modeled as a bar-and-joints framework [21]. The rigidity theory relates between structures and graphs, and

using the algebraic graph theory allow us to analyze their rigidity properties [22, 23, 24] Firstly, we shall define the notion of a configuration, which describes the positions of the agents of the system.

Definition 2.2.1 (configuration). A d-dimensional configuration is a finite collection of n points denoted $c = \begin{bmatrix} p_1^T & \cdots & p_n^T \end{bmatrix}^T \in \mathbb{R}^{n \times d}$, where $p_i \in \mathbb{R}^d$.

We also assume that $p_i \neq p_j \ \forall i \neq j$.

Next, we define a relation between a graph and the real physical world. A *framework* $\mathcal{F} = (\mathcal{G}, p)$ is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the mapping $p : \mathcal{V} \to \mathbb{R}^2$, which assigns to each node a point in a metric space [25, 26, 21, 23]. In this way, $p(v) \in \mathbb{R}^2$ is the position assigned for a graph node $v \in \mathcal{V}$, and $p(\mathcal{V}) = c$. We will often use the shorthand $p_i = p(v_i)$ to denote the position of the nodes in a framework. In our work, we focus on the Euclidean space $\mathbb{R}^2$, and the distance between two nodes is the standard Euclidean norm, $\|p_i - p_j\|^2 = (p_i - p_j)^T(p_i - p_j)$.

Now we are ready to define some important properties of frameworks. Two frameworks are said to be *equivalent* if they have the same edges length, and they are *congruent* if the distance between each pair of points is identical.

Definition 2.2.2 (equivalent and congruent frameworks). Two frameworks $\mathcal{F}_0 = (\mathcal{G}, p), \mathcal{F}_1 = (\mathcal{G}, q)$ are *equivalent* if

$$\|p(v_i) - p(v_j)\| = \|q(v_i) - q(v_j)\|, \ \forall \ \{v_i, v_j\} \in \mathcal{E}.$$

Two frameworks are *congruent* if

$$\|p(v_i) - p(v_j)\| = \|q(v_i) - q(v_j)\|, \ \forall \ v_i, v_j \in \mathcal{V}.$$

It is possible to think of a framework using a bar-and-joint interpretation where the bar lengths are assumed to remain fixed. Each bar thus represent an edge. An interesting question is whether there exists a movement of the entire framework that maintains the lengths of the bars but deforms the framework shape. If there isn't such a movement, then the framework is *rigid*, and an example is given in Figure 2.2(b). However, sometimes if we remove a bar from a rigid framework it remains rigid. The *minimally rigid framework* is a framework where any bar removed makes it non-rigid. We formalize these notions below.

Definition 2.2.3 (rigid framework). A framework $\mathcal{F}_0 = (\mathcal{G}, p)$ is *rigid* if there exists an $\epsilon > 0$ such that every framework $\mathcal{F}_1 = (\mathcal{G}, q)$ that is equivalent to $\mathcal{F}_0$ and satisfies $\|p(v) - q(v)\| < \epsilon \ \forall \ v \in \mathcal{V}$, is congruent to $\mathcal{F}_0$.

Definition 2.2.4 (minimally rigid framework). A *minimally rigid framework* is a rigid framework $\mathcal{F}_0$ such that the removal of any edge in $\mathcal{G}$ results in a non-rigid framework.

Now we can define the *rigidity function* [25]. Given a framework $\mathcal{F}(p) = (\mathcal{G}, p)$, we can define the function $g_G : \mathbb{R}^{2n} \to \mathbb{R}^{|\mathcal{E}|}$:

$$g_G(p) \triangleq \left(\cdots, \|p_i - p_j\|^2, \cdots\right)^T. \tag{2.1}$$

The *rigidity matrix* $R(p)$ associated with the framework $\mathcal{F}(p)$ is the Jacobian of the rigidity function $\frac{\partial g_G(p)}{\partial p} \in \mathbb{R}^{|\mathcal{E}| \times 2|\mathcal{V}|}$. Define $w_k = p_i - p_j$, and $w = \left[w_1^T, \ldots, w_{|\mathcal{E}|}^T\right]^T$. It can be also shown that ([27])

$$R(p) = \mathrm{diag}(w_k^T)\left(E(\mathcal{G})^T \otimes I_2\right), \tag{2.2}$$

where $\mathrm{diag}(w_k^T)$ is a $|\mathcal{E}| \times 2n$ block diagonal matrix with w_k^T on the block diagonal.

As defined above, a configuration is the embedding of a graph and the physical world. As such, it can represent dynamical situations, where the positions are not absolutely static but subject to changes. We're looking for infinitesimal motions, which keeps the rigidity function constant up to first order. A first order Taylor expansion of the rigidity function about p will be [25]

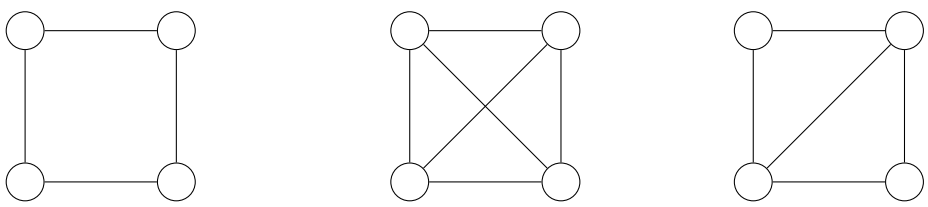
$$g_G(p + \delta p) = g_G(p) + R(p)\delta p + O(\delta p^2).$$

If $\delta p \in \ker R(p)$, $R(p)\delta p = 0$, and then we call δp an infinitesimal flex of the framework [26].

Definition 2.2.5 (infinitesimal flex). If $\exists\, dp \in \mathbb{R}^{2n}$ such that $R(p)dp = 0$, then dp is an *infinitesimal flex* of $\mathcal{G}$.

A framework is *infinitesimally rigid* if the only infinitesimal flexes are trivial, i.e., the rigid body rotations and translations of the framework. However, infinitesimal rigidity does not imply rigidity. A minimally rigid framework graph must have exactly $2n - 3$ edges [28]. A minimally rigid framework that is also infinitesimally rigid is called *minimally infinitesimally rigid* (MIR). An example is Figure 2.2(c).

Lemma 2.2.6 ([28])**.** *A framework $\mathcal{F}$ is infinitesimally rigid if and only if $rank(R(p)) = 2n - 3$.*



(a) Non rigid framework (b) Rigid & infinitesimal framework. (c) MIR framework.

Figure 2.2: Rigidity of different frameworks.

All of the above definitions help us understand what happens to a framework when perturbations are made. If the framework represents a number of agents and it is rigid then we can be assured that perturbations won't change the spatial shape (i.e. structure) of the agents. Doing so helps to maintain a required structure, or *formation*, which is very important for different tasks.

2.3 Dynamical Systems

A dynamical system is a system which a time-dependent function describes its state in a geometrical space. For this work, we consider the stability for an autonomous dynamic system described by a set of non-linear equations,

$$\dot{x} = f\left(x(t)\right), \tag{2.3}$$

where $x : \mathbb{R} \to \mathbb{R}^n$, and $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$, and t is the independent variable (usually representing time). We assume that there exists a continuous, unique solution with the initial conditions $x(t_o) = x_0 \in \mathbb{R}^n$, where t_0 is some initial time (constant). The equilibrium points of (2.3) is simply the solution for $\dot{x} = 0$, and let us mark an equilibrium of this equation as $\bar{x}$.

In the following subsections, we will firstly define and discuss what stability is, and then focus on gradient systems which have unique properties.

2.3.1 Stability Theory

The stability theory is the study of the stability of trajectories arising from dynamical systems (2.3) modeled with differential equations. A stable system is one that the solution *trajectory* doesn't change drastically when small perturbations are made to the initial conditions. The stability of the equilibrium point $\bar{x}$ can now be characterized. Let's assume we perturb system (2.3) from its equilibrium by some $\epsilon \in \mathbb{R}^n$ at the time t_0, such that $x(t_0) = \bar{x} + \epsilon$. If the trajectories remain close to the nominal equilibrium, in some sense, we say the system is stable.

Without loss of generality, we assume that $\bar{x}$ is the origin of the axes on the state-space (and if it isn't - a state transformation can be done with some new variable and a study of the new transformed system can be made). Then, we can define formally the notion of Lyapunov stability [29].

Theorem 1 (Lyapunov Stability)**.** *The equilibrium point of (2.3) is*

- *stable if for each $\epsilon > 0$ there exists $\delta > 0$ such that for any $\|x(t_0)\| < \delta$, one has $\|x(t)\| < \epsilon, \forall t > t_0$,*
- *unstable if it is not stable,*

- *and asymptotically stable if it is stable and δ can be chosen such that $\|x_0\| < \delta \Rightarrow \lim_{t\to\infty} x(t) = 0$.*

While in a general stable system the trajectories are bounded by a ball with radius ϵ, an asymptotically stable system will approach the origin, thus providing a stronger notion of stability.

Lyapunov also proposed two methods to analyze the stability of a system. The first method, named *Lyapunov's First Method* (also known as *indirect method*) is not of our interest for this work. This method analyzes the eigenvalues of the linearized dynamics of (2.3). Further information about this method can be found on [30].

Lyapunov's second method, also known as the *direct method*, does not require a characterization of the solution of (2.3). This method relies on a generalization of the enrgy, or potential, of a system. If the system has some energy, then we can use the rate of energy change to determine its stability. First of all, we should define the *candidate Lyapunov function* $V(x) : D \to \mathbb{R}$ where $V(x)$ is a continuously differentiable function defined on $D \subset \mathbb{R}^n$ that contains the origin. We are interested in the rate of change of V along the trajectories of (2.3),

$$\dot{V}(x(t)) = \frac{d}{dt}V(x(t)) = \frac{\partial V}{\partial x} \cdot \frac{dx}{dt} = \nabla V \cdot \dot{x} = \nabla V \cdot f(x). \tag{2.4}$$

The main idea of Lyapunov's theory is to establish properties of the nonlinear system by studying how certain carefully selected scalar functions of the state evolve as the system state evolves. The rate of change of $V(x)$, $\dot{V}(x)$ along any trajectory of the solution for (2.3) defines how this system evolves. When $\dot{V}(x(t))$ is negative, we can say that the potential decreases towards zero, hence the system is stable [29].

Theorem 2 (Lyapunov Direct Method)**.** *Let the origin $0 \in D \subset \mathbb{R}^n$ be an equilibrium point of (2.3). Let $V(x) : D \to \mathbb{R}$ be a continuously differentiable function satisfying*

(1) $V(0) = 0$,

(2) $V(x(t)) > 0, \forall x(t) \in D \setminus \{0\}$,

(3) $\dot{V}(x(t)) \leq 0, \forall x(t) \in D$.

Then, $x(t) = 0$ is a locally stable solution of (2.3). Moreover, if $\dot{V}(x(t)) < 0, \forall x(t) \in D \setminus 0$ then $x(t) = 0$ is locally asymptotically stable.

If the conditions in Theorem 2 are met, then V is called a Lyapunov function for the system described in (2.3). Unfortunately, Lyapunov's direct method assumes the existence of a Lyapunov function, but does not provide any method to construct one from the differential equation (2.3).

While Lyapunov methods are highly valuable, sometimes finding an appropriate Lyapunov function can be difficult. Moreover, sometimes it is possible to meet Theorem 2 conditions for stability, but it can be very difficult to prove the condition for asymptotical stability ($\dot{V}(x(t)) < 0, \forall x(t) \in D \setminus 0$ then $x(t) = 0$).

To help us with the mentioned above issues, LaSalle provided a theorem [31, 29]. For this theorem, which is an extension of Lyapunov's second method, LaSalle used the notion of limit sets and the notion of invariance (the property of certain sets whereby a given function takes elements in the set to elements in the same set). By introducing these notions, LaSalle was able to show how Lyapunov functions could be defined less restrictively.

LaSalle's theorem enables one to conclude asymptotic stability of an equilibrium point even when $\dot{V}(x(t))$ is negative semi-definite. We begin by introducing a few more definitions. We denote the solution trajectories of the autonomous system in (2.3) as $s(t, x_0, t_0)$, which is the solution at time t starting from x_0 at t_0. The ω-limit set is the set $S \subset \mathbb{R}^n$ of a trajectory $s(\cdot, x_0, t_0)$ if for every $y \in S$, there exists a strictly increasing sequence of times t_n such that $s(t_n, x_0, t_0) \to y$ as $n \to \infty$. A positive *invariance set* is the set $M \subset \mathbb{R}^n$ if for all $y \in M$ and $t_0 \geq 0$ we have $s(t, x_0, t_0) \in M \ \forall t \geq t_0$. It is also possible to prove that the ω-limit set of every trajectory is closed and invariant.

Theorem 3 (LaSalle's Invariance Principal)**.** *Let $V : \mathbb{R}^n \to \mathbb{R}$ be such that on $\Omega_c = \{x \in \mathbb{R}^n : V(x) \leq c\}$, a compact set we have $\dot{V}(x) \leq 0$. Define*

$$S = \{x \in \Omega_c : \dot{V}(x) = 0\}.$$

Then, if S contains no trajectories other then $x = 0$ then the origin is asymptotically stable.

Full proofs of these results can be found in [29, 31].

2.3.2 Gradient Dynamical Systems

Gradient dynamical systems are a unique case of dynamic systems, where the behaviour can be described as a negative gradient flow [32]. Many systems can be described as such, and gradient dynamics systems have some unique properties.

Given a differentiable function $V(x(t)) : \mathbb{R}^n \to \mathbb{R}$, the negative gradient flow defined by V is the dynamical system

$$\dot{x} = -\frac{\partial V}{\partial x}(x(t)). \tag{2.5}$$

Let us further assume that V is twice differentiable. Then, for a point $x \in \mathbb{R}^n$, the *Hessian matrix* of V at x, denoted $\mathrm{Hess}(V(x)) \in \mathbb{R}^{n \times n}$, is a symmetrical matrix of second order derivatives at x. If x^* is a critical point of V, and if $\mathrm{Hess}(V(x^*))$ is positive definite, then x^* is a local minimum of $V(x)$ [32].

As mentioned above, gradient dynamic systems have special properties, which are described in the following theorem [32].

Theorem 4 (Convergence of negative gradient flow)**.** *Let $V : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable and assume its sublevel set $V_{\leq}^{-1}(l) = \{x \in \mathbb{R}^2 \mid V(x) \leq l\}$ is compact for some $l \in \mathbb{R}$. Then, the negative gradient flow* (2.5) *has the following properties:*

(1) Each solution $x \to x(t)$ starting at $V_{\leq}^{-1}(l)$ satisfies $\lim_{t\to\infty} V(x(t)) = c$ for some $c \leq l$, and approaches the set of critical points of V:

$$\{x \in \mathbb{R}^n \mid \frac{\partial V}{\partial x}(x) = 0_n\},$$

(2) Each local minimum point x^* is locally asymptotically stable and it is locally exponentially stable if and only if Hess$(V(x^*))$ is positive definite,

(3) A critical point x^* is unstable if at least one of the eigenvalues of Hess$(V(x^*))$ is strictly negative,

(4) If the function V is analytic, then every solution starting in a compact sublevel set has a finite length (as a curve in $\mathbb{R}^n$) and converges into a single equilibrium point.

The key idea in the proof of this result is that the function V defining the dynamics serves as a natural Lyapunov function for the system. For the full proof, see [32].

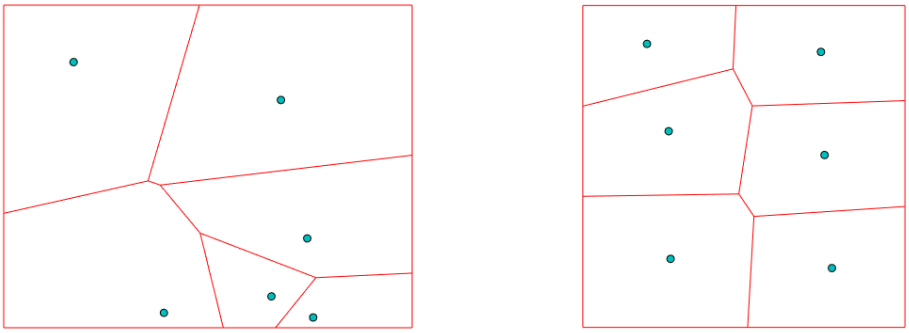
2.4 Voronoi Tessellations

The basic mathematical concept that we employ in this work is the *Voronoi Diagram* (also known as Voronoi partition or Voronoi tessellation). While being a method to partition an area with some cost function, it is a widely used in the optimal coverage problems [9, 10, 13].

The Voronoi Diagram of a region $A \subset \mathbb{R}^2$ is the set of partitions $\mathcal{V} = \{V_1, \ldots, V_n\}$ generated by the generators $\mathcal{Z} = \{z_1, \ldots, z_n \mid z_i \in A\}$, such that

$$V_i = \{q \in A \mid \|q - z_i\| \leq \|q - z_j\| \; \forall \; i \neq j\}, \tag{2.6}$$

and $\cup_i V_i = A$, where $\|\cdot\|$ denotes the Euclidean distance. A more intuitive and non-formal definition of the Voronoi Diagram is as follows. If we take some area and place points p_i in it, then each partition is the set of all points that are closer to p_i than p_j, when $j \neq i$. An example is the problem where there are post offices in some city, and it is needed to decide which house will be served by which specific post office. Using Voronoi partitioning, we can determine which houses are the nearest to each office and therefore conclude which houses each branch will serve. A visual example can be seen in Figure 2.3(a).



(a) Voronoi Diagram.

(b) Centroidal Voronoi Tessellations. Calculated using the density function $\rho = 1$.

Figure 2.3: Illustration of Voronoi Diagram and a CVT. The dots are the cells' generators.

One can also define a density function, ρ_i, for each Voronoi partition V_i. Then, we can define the center of mass for each partition as

$$z_i^* = \frac{\int_{V_i} y\rho(y)dy}{\int_{V_i} \rho(y)dy}. \tag{2.7}$$

If a generator $z_i = z_i^* \, \forall \, V_i$, we call this partitioning a *centroidal Voronoi tessellation* (CVT). Such tessellations are useful in terms of location optimization [9, 13, 6]. A centroidal Voronoi tessellation illustration can be seen in Figure 2.3(b).

Remark. Throughout this work, $\rho = 1$ was used.

While Voroni diagrams are useful for partitioning some existing range, CVT is a technique used for (but not only) *planning* in an optimal way. If in the Voronoi diagram explanation, it was used to plan the delivery areas for each office, we will use the CVT to find where are the optimal locations to place the post offices.

Calculating the CVT might be a complicated task. Lloyd proposed a very simple way of calculating the CVT [14], presented in Algorithm 2.1. Various analysis were made over Lloyd's algorithm convergence for the particular use of CVT calculations [33].

There are many diverse real-world applications for Voronoi partitioning. One that was already given talks about postal-service. Another application, which is more relevant for our case, is the optimal coverage strategy for given agents and an area. If we assume that the generators represent some agents, or sensors, in some way the tessellations are the optimal area for the corresponding agent to cover. While the Voronoi tessellations provides an optimization for the current state, the CVT can provide us with an optimal *deployment* for the system.

Algorithm 2.1 Lloyd's Algorithm

Input: $A \subset \mathbb{R}^2,\ \rho_i, p_0$ ▷ $p_0 = p(t_0)$ is the initial agents positions
Output: z_i^*
1: **while** $z_{i_k} - z_{i_{k-1}} > \epsilon$ **do**
2: Calculate the Voronoi diagram for A, using generators $z_{i_{k-1}}$. ▷ For first iteration, use p_0 as initial guess.
3: $C_{V_i} \leftarrow$ Calculate the center of mass for every cell using $\rho_i(y)$.
4: $z_{i_k} \leftarrow C_{V_i}$.
5: **end while**
6: $z_i^* \leftarrow z_{i_k}$

2.4.1 Lloyd's Algorithm Based Deployment Control

Cortes et al. [9] proposed a deployment control algorithm based on Lloyd's algorithm. Let us define a polygon $\mathcal{W} \subset \mathbb{R}^2$. The basic deployment problem is finding a position $p \in \mathbb{R}^2$ to maximize the coverage of the polygon $\mathcal{W}$. According to [9], and assuming the density function $\rho(y) = 1$, minimizing the function

$$\mathcal{H}(p, \mathcal{W}) = \int_{\mathcal{W}} \|p - q\|^2 dq, \tag{2.8}$$

will yield the requested result. The function $\mathcal{H}(p, \mathcal{W})$ is the locational optimization function. In [9], this problem was expanded to the case of n points, such that each point p_i represent the i'th sensor position, and we are looking for the best partition $(\mathcal{W}_1, \ldots, \mathcal{W}_n)$ such that the coverage is maximized. Now, the total cost function will be

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{\mathcal{W}_i} \|p_i - q\|^2 dq. \tag{2.9}$$

Minimizing this function results with the Voronoi partitioning, marked as $\mathcal{V} = (V_1, \ldots, V_n)$.

The main challenge now is finding the points that minimize (2.9). According to [9], when using Voronoi partitioning, (2.9) can be expressed as:

$$\mathcal{H}(P, \mathcal{V}) = \mathcal{H}_{\mathcal{V}}(P) = \int_A \min_{i \in (1,\ldots,n)} \|p_i - q\|^2 \phi(q) dq, \tag{2.10}$$

where $\phi(q)$ represents a measure of information or probability that some event take place over A. For the rest of the problem, and for the sake of simplicity, we assume that $\phi(q) = \rho(q) = 1$.

It is possible to define the mass M_{V_i}, moment of inertia J_{V_i} and the center of mass C_{V_i} of the i'th Voronoi region as,

$$M_{V_i} = \int_{V_i} \rho(q) dq, \tag{2.11}$$

$$J_{V_i} = \int_{V_i} \|q - p_i\|^2 \rho(q) dq, \tag{2.12}$$

$$C_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} q\rho(q)dq. \tag{2.13}$$

Cortes et al. [9] showed that at least one of the solutions for the optimization of (2.10) is the CVT,

$$C_{V_i} = \arg\min_{p_i} \mathcal{H}_{\mathcal{V}}(p). \tag{2.14}$$

Note that there might be more then one solution for this problem and the selected one is dependent on the initial conditions.

Assuming that a sensor has integrator dynamics, i.e., for each sensor, $\dot{p}_i = u_i$, [9] shows a way to implement Lloyd's algorithm as,

$$u_i = -k_p \left(p_i - C_{V_i}\right), \tag{2.15}$$

where k_p is a positive constant, and C_{V_i} is recalculated at each time step. Note that the dynamics in (2.15), which minimize (2.8), is locally asymptotically stable [9]. In [9, 19], it's shown that $u_i = -\frac{\partial \mathcal{H}_{\mathcal{V}}(P(t))}{\partial P}$, thus system (2.15) is a gradient flow system. moving the agents along using controller (2.15) results with a trajectory leading to optimal deployment.

Remark. For Figure 2.3(b) and throughout this work, $k_p = 5$ and $\rho = 1$ were used.

2.5 Formation Control

Acquiring and maintaining a spatial structure of several agents that should cooperate for achieving some goal is a difficult mission. In this chapter, we will show how previously defined mathematical tools such as graph theory, rigidity theory and stability theory can help us define controllers that solves this problem.

In this work, we will focus on *distance based formation control*. Consider a system of $n \geq 2$ agents, moving in $\mathbb{R}^2$. The agents have single integrator dynamics,

$$\dot{p}_i = u_i, i = 1, \ldots, n, \tag{2.16}$$

where $p_i(t) \in \mathbb{R}^2$ is the coordinate vector assigned to the i'th agent, and $u_i \in \mathbb{R}^2$ is the control input associated to each agent. Those agents can also communicate one with each other, sharing relative measurements (location, relative distance, etc.).

Graphs are a natural mathematical tool for describing formations [15]. The desired distance between two agents i, j connected by k'th edge is d_k (sometimes will be marked as d_{ij}), and let $d = \begin{bmatrix} d_1^2 & \ldots & d_m^2 \end{bmatrix}^T \in \mathbb{R}^{|\mathcal{E}|}$ be the required distance vector ($m = |\mathcal{E}|$). The *distance error* is the vector that measures the square of the difference between the measured distance and the desired distance,

$$\delta_k = \|w_k\|^2 - d_k^2, \; k \in \{1, \ldots, m\}, \tag{2.17}$$

where w_k is the relative positions between two agents. A proposed control algorithm should achieve the following asymptotic result, solving the formation control problem,

$$\lim_{t\to\infty} \|p_i - p_j\| = \lim_{t\to\infty} \|w_k\| = d_k,\ k \in \{1,\dots,m\}. \tag{2.18}$$

Equation (2.18) can be equivalently written in terms of the distance error,

$$\lim_{t\to\infty} \|\delta_k\| = 0,\ k \in \{1,\dots,m\}. \tag{2.19}$$

In [15], a locally asymptotically stable gradient controller was proposed. It was based on the following potential function:

$$F(p) = \frac{1}{4}\sum_{k=1}^{m}\left(\|w_k\|^2 - d_k^2\right)^2 = \frac{1}{4}\sum_{k=1}^{m}\delta_k^2. \tag{2.20}$$

Note that this potential function zeroes only when for each agent, $\|w_k\|^2 = d_k^2$. Therefore, it also represents the total formation error and will be sometimes marked also as δ. The proposed controller is the negative gradient flow of (2.20),

$$\dot{p} = -\frac{\partial F(p)}{\partial p}.$$

For each agent k, the control law is of the form

$$\dot{p}_i = u_i = -\sum_{i\sim j}\left(\|p_j - p_i\|^2 - d_{ij}^2\right)(p_i - p_j). \tag{2.21}$$

Using the framework's graph rigidity matrix, the same control law can be written as

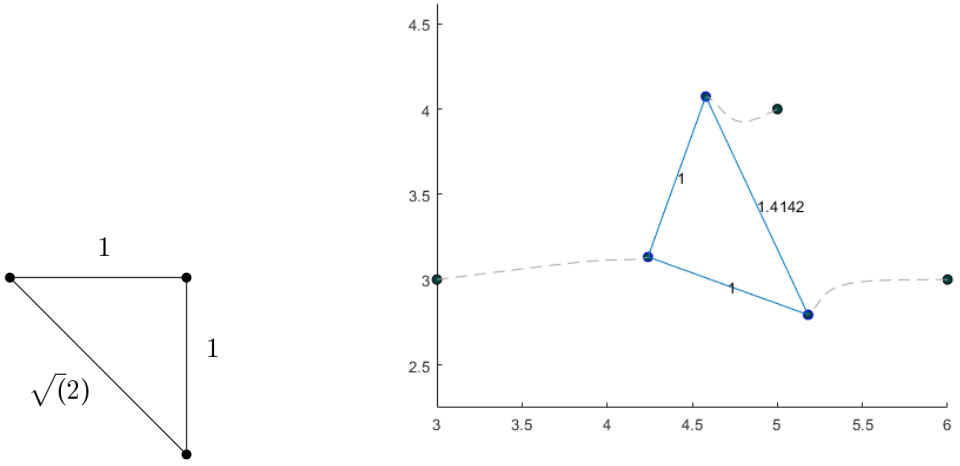
$$\dot{p}(t) = u = -R(p)^T R(p)p - R(p)^T d^2. \tag{2.22}$$

Where d^2 is a vector with its elements squared (i.e. $d^2 = \begin{bmatrix} d_1^2 & \cdots & d_m^2 \end{bmatrix}^T$). Controller (2.22) has an equilibrium point exactly where we want it to be:

$$\dot{p}(t) = 0 \Rightarrow -R(p)p - d^2 = 0 \Rightarrow d_k^2 = \|w_k\|^2. \tag{2.23}$$

An example for this controller can be seen on Figure 2.4. The required formation is a right triangle, with edge lengths goal distances as $1, 1, \sqrt{2}$.

As for stability, different stability analysis can be found in [15, 25, 34]. Generally, the distance-based formation control is locally asymptotically stable if the underlying graph is at least MIR. The proof is based on the fact that the controller has gradient dynamics, and given on Subsection 2.3.2.



(a) Required Formation.

(b) Simulation showing initial position (black), trajectory, and final position (blue) of agents. The numbers represents the distance between the agents.

Figure 2.4: An example for classic distance-based formation control.

Chapter 3

Problem Formulation

This chapter formalizes the definitions and problem statement of this thesis. First, it will describe the sensor *coverage* of an area with limited amount of sensors and sensing constraints. Secondly, it will focus on formation maintaining while performing the coverage mission.

3.1 Constrained coverage

Let us consider an area $A \subset \mathbb{R}^2$ that we aim to cover with $n \in \mathbb{N}$ *mobile* sensors. Each sensor is labeled as s_i, and the set of all the sensors is given by $S = \{s_1, \ldots, s_n\}$. We further associate with each sensor s_i a position, $p_i(t) \in \mathbb{R}^2$. The mobile sensors are modeled with integrator dynamics, i.e.,

$$\dot{p}_i(t) = u_i,\ i = 1, \ldots, n.$$

The configuration of the collection of sensors at time t is given by the vector $c(t) = \left[p_1^T(t) \quad \cdots \quad p_n^T(t)\right]^T$. Each sensor can cover a region described by the abstract area $C_i\left(p_i(t)\right) \subset \mathbb{R}^2$, as can be seen in Figure 3.1.

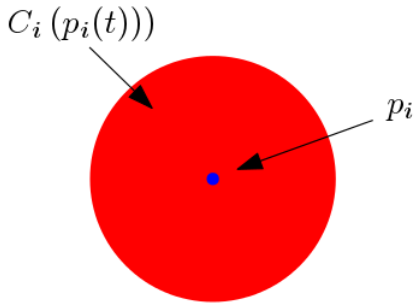


Figure 3.1: A sensor and its coverage region.

Thus, the total coverage of a configuration is given by

$$D\left(c\left(t\right)\right) = \cup C_i(p_i(t)).$$

We also emphasize in this work that we assume it is not possible to cover the entire area A using a single configuration. That is, $A \not\subset D(c(t))$ for any configuration $c(t)$.

Assumption 1. For any configuration $c(t)$, the coverage satisfies $A \not\subset D(c(t))$.

Sometimes we would like to maintain some area under partial coverage at all time, for example, a ground station which must maintain communication with the sensors. In the case where it's needed, let us consider a sub-area of A, $A_m \subset A$. The partial coverage constraint can be described as the following,

$$A_m \cap D\left(c(t)\right) \neq \emptyset. \tag{3.1}$$

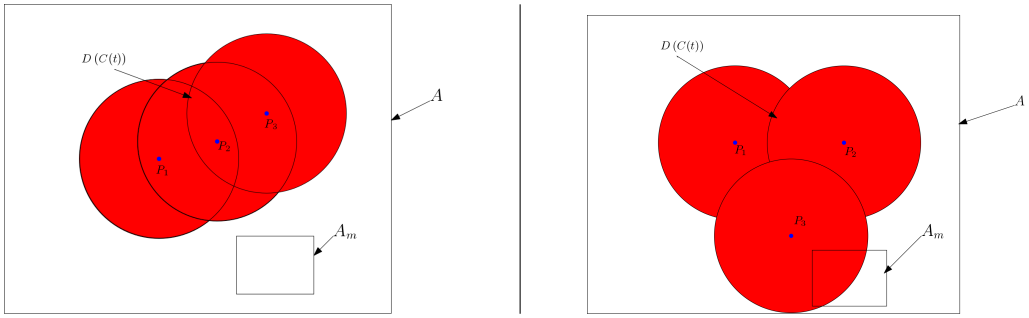


Figure 3.2: Two configurations built from the same sensors set.

In Figure 3.2, we can see how a given configuration can cover some area (assuming disk coverage for each sensor) in different ways. In one case, A_m is partially covered and in the other it isn't. Due to Assumption 1 we know that it isn't possible to achieve full area coverage using a single configuration. Therefore, we shall *partition* the area, as can be seen in Figure 3.3.

Definition 3.1.1 (Partition). A *partition* of $A \in \mathbb{R}^2$, denoted $PR(A)$, is a finite set built from l subsets $pr_i \subset \mathbb{R}^2$, $i = 1, \ldots, l$ such that

i) $pr_i \cap pr_j = \emptyset \; \forall i \neq j$.

ii) $\cup pr_i = A$.

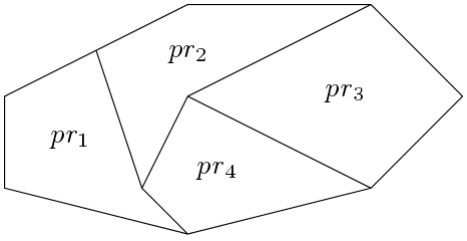


Figure 3.3: An example of a partition of some area.

So far, we've defined sensor coverage, configurations and partitioning. Following Assumption 1 and Condition (3.1), we have to find a partition $PR(A)$ such that each of it's subsets pr_i intersects with A_m. This will result with set of tiles that the union of them provides full coverage of A and will fulfil Condition (3.1). Then, our strategy will require covering each tile on it's own, and by covering *each tile on it's own* by the sensors, eventually, using some strategy (sequential, for example), we will eventually reach full coverage. One other major assumption that is hidden is that it is possible to cover each pr_i using our set of sensors.

Problem 1 (Constrained Coverage) Given a set of mobile sensors with integrator dynamics, an area A and sub-area constraint A_m, and under Assumption 1,

i) Find a l-partition $PR(A)$ such that each of the subsets $pr_i \in PR(A)$ satisfy $pr_i \cap A_m \neq \emptyset$;

ii) Find a deployment strategy for each partition, such that there exists a sequence of times T_i satisfying $T_\ell > T_{\ell-1} > \cdots > T_1 > t_o$ satisfying

$$pr_i \subseteq D(c(T_i))), \text{for } \; i = 1, \ldots, l.$$

Condition (ii) states that each partition must be covered by the deployment strategy after some finite time. While condition (i) of the problem is understandable, the idea behind condition (ii) is that full coverage of A under condition (3.1) is possible if we perform coverage control on each partition pr_i. An example for partitioning with the constraint (3.1) can be seen in Figure 3.4.

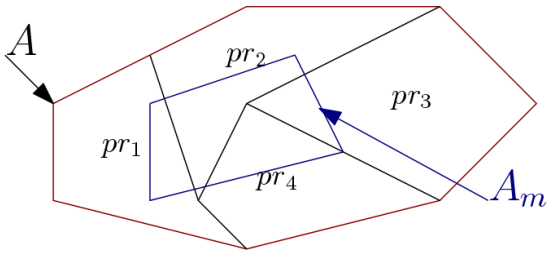


Figure 3.4: A solution example for Problem 1.

3.2 Formation Control and Coverage Control

Let us consider a configuration $c(t)$ and a partition pr_i as defined on Definition 3.1.1. For a configuration $c(t)$, we define an underlying connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which an edge $e_{ij} \in \mathcal{E}$ represents communication between nodes $v_i, v_j \in \mathcal{V}$, and the pair $(\mathcal{G}, c)$ defines a *framework*.

In a framework, the *edge length* represents the distance between two agents. In Section 2.5 we discussed formation control, and established that in order to maintain

formation the graph $\mathcal{G}$ has to be infinitesimally rigid. The set $d \in \mathbb{R}^{|\mathcal{E}|}$ is the set of required distances between agents (that is, assuming $\{v_i, v_j\} \in \mathcal{E}$, then d_{ij} is the required distance between agents i, j).

In this problem, we aim to solve Problem 1 under and addition a formation constraint defined by d.

Problem 2 (Coverage and Formation). Given a framework $\mathcal{F} = (\mathcal{G}, c)$, such that $\mathcal{F}$ is infinitesimally rigid, and given the desired formation specified by the distance vector d,

i) Find a partition $PR(A)$ such that each of the subsets $pr_i \in PR(A)$ satisfy $pr_i \cap A_m \neq \emptyset$;

ii) Find a deployment strategy for each partition, such that there exists a sequence of times T_i satisfying $T_\ell > T_{\ell-1} > \cdots > T_1 > t_o$ satisfying $pr_i \subseteq D(c(T_i)), \text{for } i = 1, \ldots, l$, while satisfying $\lim_{t_o \to T} \|w_k\| = d_k + \epsilon$ (where ϵ is a constant).

The idea behind Problem 2 is using the partitioning coverage strategy, obtaining a required formation on each partition. Each partition defined on Problem 1 is answering the sub-area constraint, and now we modify the solution to include maintaining a formation. That said, in the general case the formation and deployment optimizations may not converge to the desired spatial formation, therefore we define some constant ϵ which defines an allowable formation error.

Chapter 4

The Projected Lloyd's Algorithm

In this chapter, we propose a new algorithm based on the Lloyd's Algorithm. Its goal is partitioning an area with some sub-area constraint, as mentioned in Problem 1. After the Projected Lloyd's Algorithm is presented, a full solution for Problem 1 is presented, using a centralized control strategy and partitioning strategies. Lastly, simulations are shown to demonstrate the algorithm.

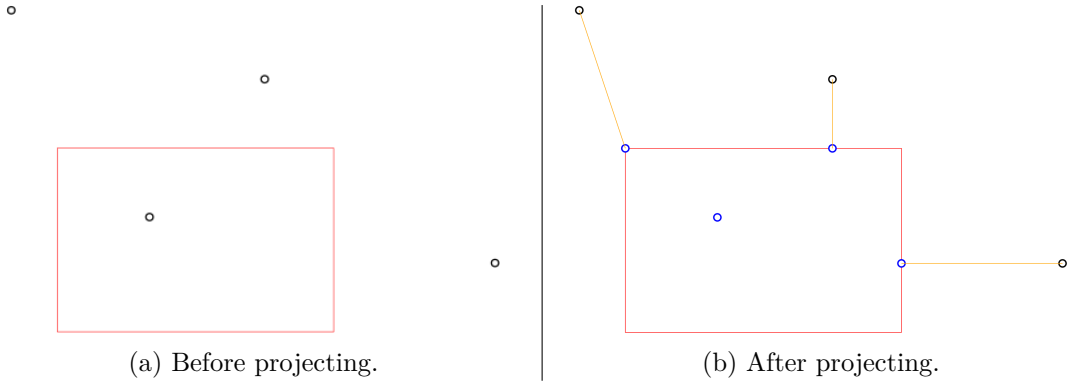
4.1 Projected Lloyd's Algorithm

In this section, a solution for Problem 1(a) under Assumption 1 will be given. Recall that in Problem 1(a), it is required that for the partition of A, $pr_i \cap A_m \neq \emptyset$ is satisfied. To show the solution, we will first define the notion of a *projection* onto the area A_m.

Projection, for this problem, will be defined as the smallest distance from a point onto a given polygon (including its boundary), and an example can be seen on Figure 4.1.

Definition 4.1.1. Consider a set $A_m \subset \mathbb{R}^2$ and a point $x \in \mathbb{R}^2$. The *projection* of x onto A_m is given by

$$\text{PROJ}_{A_m}(x) = \arg\min_{y \in A_m} \|x - y\|^2.$$



(a) Before projecting. (b) After projecting.

Figure 4.1: An example of the projection operator.

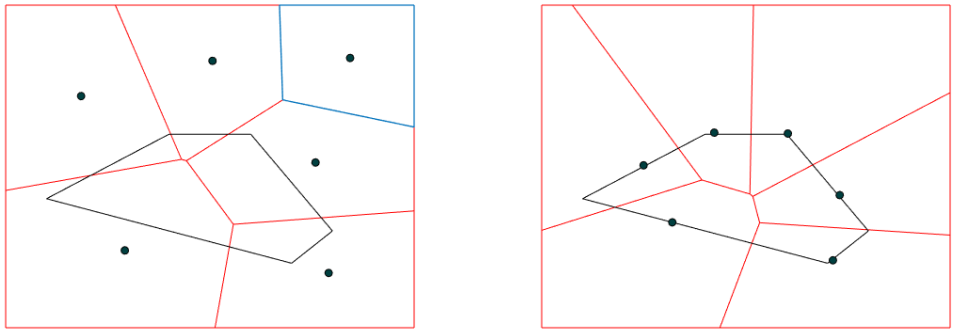
Applying the projection operator in conjunction with Lloyd's algorithm yields the *Projected Lloyd's Algorithm* (PLA), given in Algorithm 4.1. This combination allows solving Problem 1(a), and it is one of this work's main contributions.

Algorithm 4.1 Projected Lloyd's Algorithm

Input: $A \subset \mathbb{R}^2$ $A_m \subset \mathbb{R}^2$, ρ_i, p_0 ▷ $p_0 = p(t_0)$ is the initial agents positions
Output: z_i^*

1: **while** $z_{i_k} - z_{i_{k-1}} > \epsilon$ **do**
2: Calculate the Voronoi diagram for A, using $z_{i_{k-1}}$. ▷ For first iteration, use random initial guess, such as on the edges of A_m.
3: $C_{V_i} \leftarrow$ Calculate the center of mass for every cell using density functionρ_i.
4: $C_{P_i} \leftarrow \text{PROJ}_{A_m}(C_{V_i})$
5: $z_{i_k} \leftarrow C_{P_i}$.
6: **end while**
7: $z_i^* \leftarrow z_{i_k}$

As one can see, one more step was added to the original Lloyd's algorithm - where we project the cell center of mass to the polygon that defines the area constraint (step number 4). Step number 5 is modified from the original algorithm, so the agents are moving to the projected points and not the center of mass. Those steps allow us to ensure that for each partition, the agent will be on or inside the limiting polygon, thus we will have coverage within the constraint. In [33], a full convergence analysis was done for Lloyd's algorithm, however We did not perform one for the projected Lloyd's algorithm presented here. That said, our simulation studies demonstrated in practice good convergence properties. The similarity to the original Lloyd's algorithm (Algorithm 2.1) make us suspect that the proof is very similar - Eventually, this is a variation of Lloyd's Algorithm, where we constraint the center of mass to some transformation.



(a) CVT calculated using Lloyd's algorithm. On top right corner - a cell that doesn't intersects A_m.

(b) PLA solution.

Figure 4.2: CVT and PLA solutions for initial conditions. The black polygon represents A_m, and the dots are the cells' generators.

In Figure 4.2, the PLA results can be clearly seen. For the same initial conditions, Figure 4.2(a), it can be clearly seen that only 3 (out of 6 tiles) intersects with A_m. In Figure 4.2(b), the PLA algorithm was activated and all tiles intersect with A_m.

Next, a continuous time controller is proposed for implementing Algorithm 4.1,

$$u_i = -k_p \left(p_i - \text{PROJ}_{A_m} \left(C_{V_i} \right) \right). \tag{4.1}$$

This continuous time controller, which is a modification of the controller proposed in [9], is always converging into a solution where each tile intersects with A_m, if C_{V_i} is known.

In [9], it was proven that the controller (2.15) converges asymptotically to a set of critical points. It was done using (2.10) as a Lyapunov candidate function, and then, using La'Salle's invariance principal, it was shown that the sensors converge to C_{V_i}. We employ this same idea to show the convergence of the projected Lloyd's algorithm.

Theorem 5. *The controller proposed in (4.1) is locally asymptotically stable.*

Proof. Let us define a Voronoi partition moment of inertia as

$$J_{V,p} = \int_V \|q - p\|^2 \rho(q) dq. \tag{4.2}$$

In [9] it was shown that

$$\mathcal{H}_{\mathcal{V}}(P) = \sum_{i=1}^{n} J_{V_i, C_{V_i}} + \sum_{i=1}^{n} M_{V_i} \|p_i - C_{V_i}\|^2, \tag{4.3}$$

$$\frac{\partial \mathcal{H}_{\mathcal{V}}}{\partial p_i} = 2 M_{V_i} \left(p_i - C_{V_i} \right), \tag{4.4}$$

Where M_{V_i} and C_{V_i} are defined in (2.11), (2.13) respectively. Also, (4.3) is equivalent to (2.10).

Let us propose a candidate Lyapunov function:

$$\mathcal{H}_{\mathcal{P}}(P) = \sum_{i=1}^{n} J_{V_i,C_{V_i}} + \sum_{i=1}^{n} M_{V_i} \|p_i - \text{PROJ}_{A_m}(C_{V_i})\|^2, \tag{4.5}$$

it follows that [9]

$$\frac{\partial \mathcal{H}_{\mathcal{P}}}{\partial p_i} = 2M_{V_i}\left(p_i - \text{PROJ}_{A_m}(C_{V_i})\right). \tag{4.6}$$

Proof follows the Lyapunov direct method condition as stated on Theorem 2. Conditions 1,2 from the theorem are direct from the definition of $\mathcal{H}_{\mathcal{P}}(P)$. As for condition 3, it is possible to see [9] that

$$\frac{d\mathcal{H}_{\mathcal{P}}}{dt} = \frac{\partial \mathcal{H}_{\mathcal{P}}}{\partial p_i}\dot{p_i} = -2k_p \sum_{i=1}^{n} M_{V_i} \|p_i - \text{PROJ}_{A_m}(C_{V_i})\|^2 \leq 0.$$

However, the only trajectory zeroing the above equation is the trivial trajectory (i.e. $p_i = \text{PROJ}_{A_m}(C_{v_i})$), therefore by Lasalle's invariance principal, this system is locally asymptotically stable. ■

Note that both the PLA and Lloyd's algorithm converge to some local minima of the cost function. Lloyd's algorithm will converge to the CVT, but the PLA converges to some other tessellation determined by the projection onto the designated area A_m.

4.1.1 Comparing CVT and PLA Results

We now discuss the differences between the CVT and PLA algorithms. In Figure 4.2, one can see the differences between PLA and CVT results. While for the CVT we have a cell which does not intersects with A_m, the PLA calculation results with all cells having a non-empty intersection with A_m. We can also see that in the PLA case, all of the generators are *on* A_m.In (2.9), (4.5) we can see the cost function of the CVT and PLA calculations, respectively. In Figure 4.3 we can see the values of the potential function during the algorithm convergence process. While the PLA potential function values are higher, we can see that the function value is decreasing as the time advances, as expected. In Figure 4.4 we can see the resulted diagrams for which we calculated the potential. In this figure, the red square represents the area A, and while the initial positions of the agents are outside of it, the algorithm attracts it in.

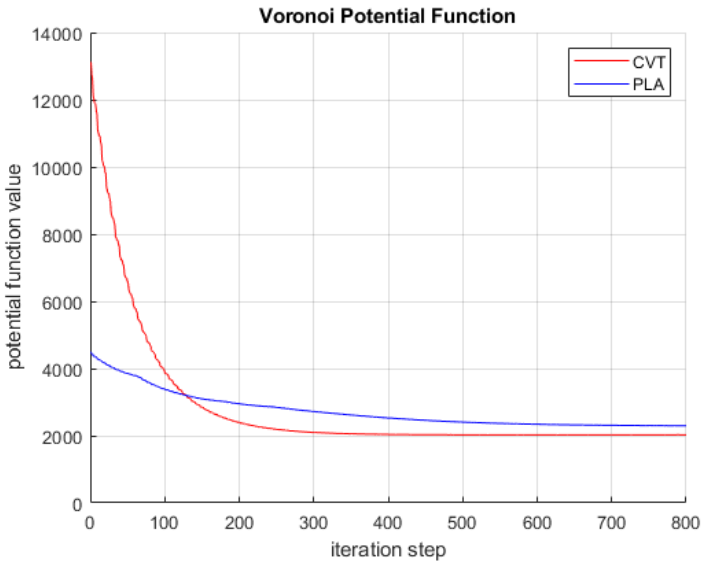
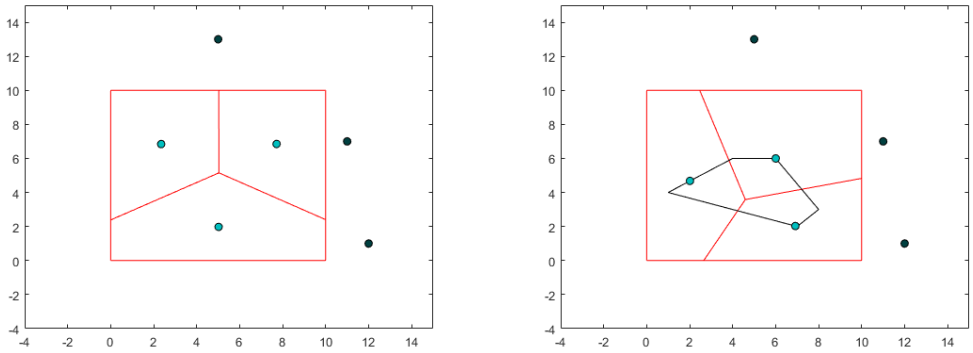


Figure 4.3: CVT and PLA potential function values for same initial conditions.



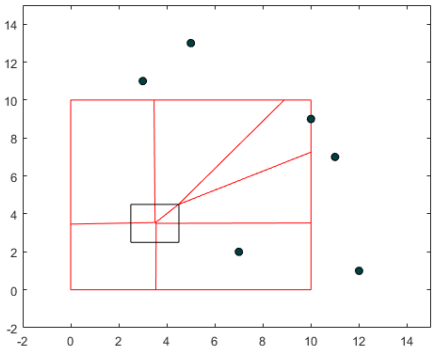
(a) Resulted CVT.

(b) Resulted PLA.

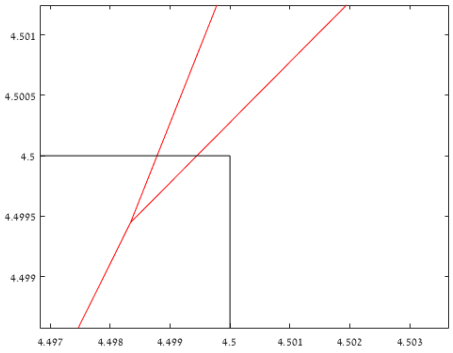
Figure 4.4: The resulted CVT and PLA diagram for the potential calculation results displayed in 4.3. Black dots represents the initial guess, turquoise the center of each cell. The black polygon in the PLA diagram represents A_m.

On Figure 4.5 we can see the impact of the initial condition of the PLA calculation, which will be described qualitatively. As the center of mass of each Voronoi cell is projected onto A_m, when the agents initial positions are on one side of it (i.e. most of the agents are to the right or left of it) - more agents will be projected onto this side of A_m. In turn, the PLA will create more cells on this side. On Figure 4.5(a) we can see the results where the initial conditions are denser towards the right, and Figure 4.5(b) is zoomed in on the upper right triangular partition (showing that the partition does intersects A_m). On Figure 4.5(c) the conditions are denser towards the left and (d) shows denser conditions towards the right.

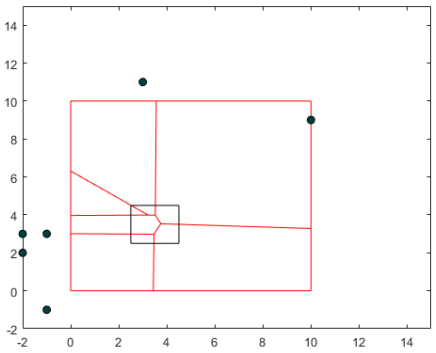
In Figure 4.6 we can see the CVT behaviour under different initial conditions. It is clear that the resulting partition is largely dependent on the initial conditions.



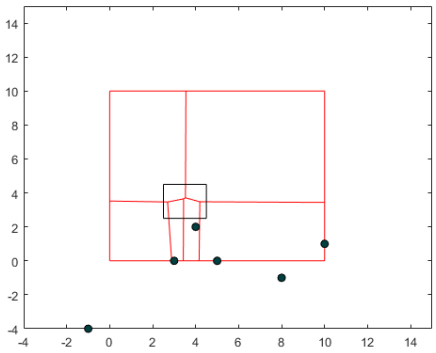
(a) PLA results with initial conditions dense to the right.



(b) PLA results with initial conditions dense to the right, zoomed in.

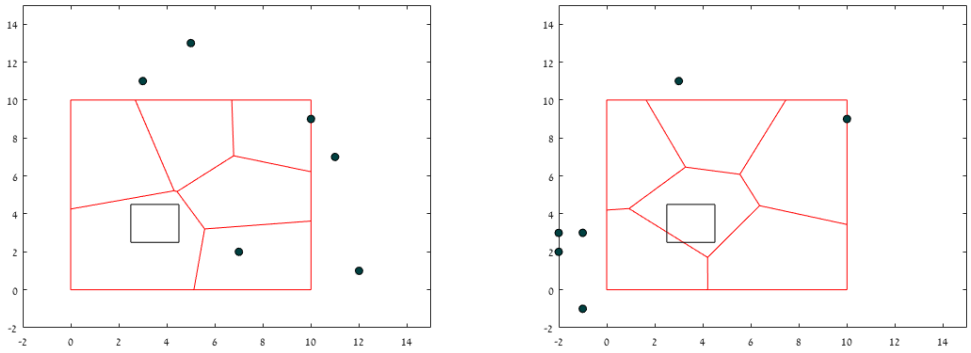


(c) PLA results with initial conditions dense to the left.



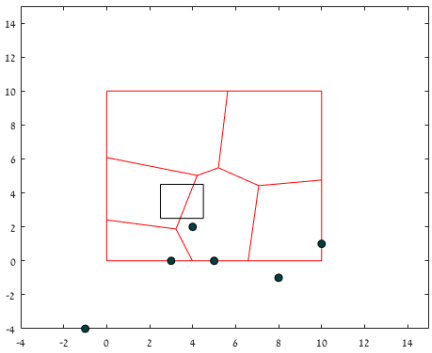
(d) PLA results with initial conditions dense to the bottom.

Figure 4.5: PLA solutions for different initial conditions (marked as black dots). The black box represents A_m.



(a) CVT results with initial conditions dense to the right.

(b) CVT results with initial conditions dense to the left.



(c) CVT results with initial conditions dense to the bottom.

Figure 4.6: CVT solutions for different initial conditions (marked as black dots). The black square represents A_m.

4.2 Problem 1 Solution Algorithm

Problem 1 consists of two sub-problems requiring to cover an area while maintaining partial coverage of some subset of the region. Problem 1 also defined partitioning an area, and for solving it we employ a sequential coverage strategy. A sequential coverage strategy means that we cover a single partition at a steady state time, and by *sequentially* covering the partitions, we achieve full coverage. As each partition intersects with A_m (as Problem 1 defined), we assure at least partial coverage of it on each partition.

Problem 1 defines conditions that must be fulfilled. First of all, we need to partition A such that $pr_i \cap A_m \neq \emptyset$, and this is where PLA algorithm comes in handy. Next question is, given a partition, we need a deployment controller to know how to cover it. This controller is presented on [9], and also in this work (2.15). Combining the PLA algorithm together with (2.15) provides us a solution to Problem 1.

Algorithm 4.2 is the proposed solution to Problem 1. The algorithm first partitions

the area to maintain partial coverage of A_m, assigns an arbitrary order to the partitions, and then performs the deployment strategy on each partition in a sequential manner according to the ordering.

Algorithm 4.2 Problem 1 Solution Algorithm

Input: $A \subset \mathbb{R}^2,\ A_m \subset \mathbb{R}^2\ , \rho_i,\ p_0,\ \ell$ partitions ▷ $p_0 = p(t_o)$

1: Calculate partitioning $PR(A)$ satisfying the sub area A_m constraint, using the PLA algorithm (Algorithm 4.1).
2: Assign an ordering to the partitions created in the previous step.
3: **for** i = 1:ℓ **do**
4: Calculate C_{V_i} using Algorithm 2.1. ▷ use current position as p_0 for Algorithm 2.1
5: Move agents pr_i according to controller (2.15)
6: **end for**

The *centralized* algorithm 4.2 proposes a simple method to solve Problem 1. First, start by partitioning the area using Algorithm 4.1. This stage promises a partitioning of A while maintaining $pr_i \cap A_m \neq \emptyset$. After it, calculating the optimal deployment for each partition using Algorithm 2.1, using the deployment of the previous partition as the initial guess for the next partition deployment calculation. Also, the Lloyd's algorithm controller (2.15) is asymptotically stable, therefore the calculation stops after a specified tolerance is achieved. *Assuming* that the sensors can cover each partition using this deployment, the Problem 1 is solved. In this work, however, we do not provide an algorithm that takes the actual sensors coverage radius into consideration when calculating the partitions.

We must notice that in stage 1, the number of the tiles of the PLA solution is decided by the user (and does not take into account $D(c(t))$). Therefore, for the assumption in step 2 to hold, the user should wisely select the number of tiles, such that the agents can provide full coverage to each tile using the optimal deployment calculated in step 2.

4.3 Numerical Results

In this sub-section, we will show results for Problem 1, using Algorithm 4.2. In the following simulations, the initial conditions were identical. In Figures 4.7, 4.8, we are simulating 3 agents covering 3 partitions using different algorithms. In each sub-figure, the black dots represents the initial position of the agents, trying to cover the "active" partition, marked in blue. The turquoise dots represents the agents final position for coveting the "active" partition. The red square represents the area A (partitioned) and the black square represents A_m. For the following simulations, we used $k_p = 5$ and $\rho = 1$.

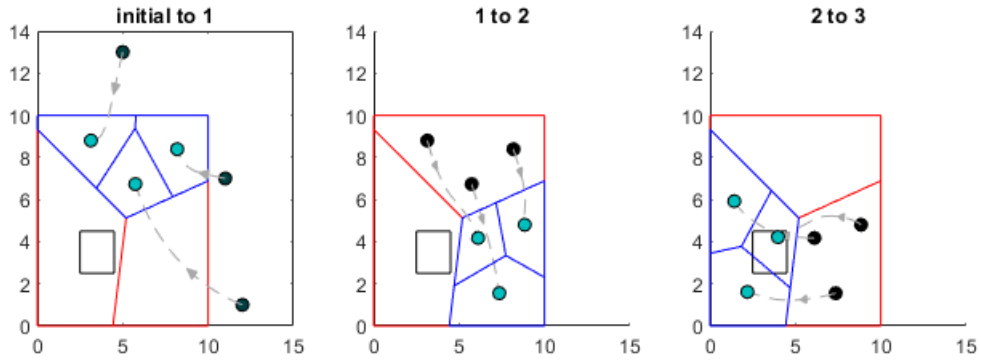


Figure 4.7: Results without PLA. The black square represents A_m.

In Figure 4.7, we can see a simulation for 3 agents[1]. The area is partitioned into 3 tiles using Lloyd's algorithm (that is CVT), and *not* using PLA, thus *not* satisfying the sub-area A_m constraint and each tile is covered using the algorithm proposed in [9]. The agents moved between the tiles in user-determined order - $1 \rightarrow 2 \rightarrow 3$, as can be seen in the figure (active partition marked in blue). One can also notice that only in the last step there was coverage of A_m.

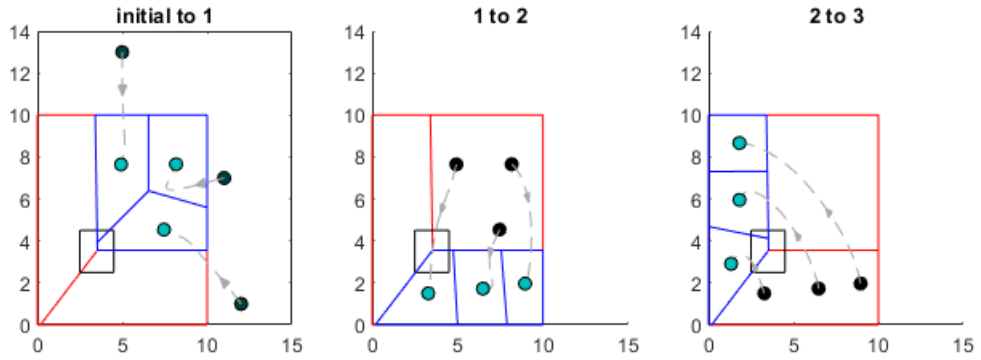


Figure 4.8: Results with PLA. The black square represents A_m.

In Figure 4.8, we perform another simulation with 3 agents. This time, the area is partitioned into 3 tiles *using* the Projected Lloyd's algorithm[2], and each tile is covered using the algorithm proposed in [9]. The agents moved between the tiles in an order to what was defined in the previous simulation ($1 \rightarrow 2 \rightarrow 3$), and each tile has some partial coverage of A_m, as expected by the algorithm.

[1]Voronoi diagram calculation here and throughout this work is done using Hyongju Park (2019). Polytope bounded Voronoi diagram in 2D and 3D (https://www.github.com/hyongju/Polytope-bounded-Voronoi-diagram), GitHub.

[2]During this entire work, the projection calculation is done using a modified version of the "p_poly_dist" Matlab function by Michael Yoshpe, https://www.mathworks.com/matlabcentral/fileexchange/12744-distance-from-points-to-polyline-or-polygon.

Chapter 5

Lloyd's Algorithm and Formation Control

Locating an emitter is an example for a mission that requires not only area coverage, but also specific spatial configuration (or, a *formation* [15]) - different configurations can affect dramatically the locating algorithm performance [35]. This problem is defined as Problem 2, and to solve this problem, the *distance-based* formation control will be combined with a deployment controller.

5.1 Formation and Deployment Control

In this section, a combined deployment and formation controller will be shown. Recall that both the deployment and formation control algorithms given in (2.15) and (2.22) are gradient dynamical systems. Our strategy is to combine them into a single potential function.

Let us define a coefficient $0 \leq \alpha \leq 1$. Thus, we propose the combined potential function as the convex combination between (2.9) and (2.20),

$$\begin{aligned}\mathcal{H}_C(p) &= \alpha\mathcal{H}(P,\mathcal{V}) + (1-\alpha)\,F(p) \\ &= \alpha\left(\int_A \min_{i\in(1,\ldots,l)} \|p_i - q\|^2\phi(q)dq\right) + (1-\alpha)\left(\frac{1}{4}\sum_{k=1}^{m}\left(\|w_k\|^2 - d_k^2\right)^2\right),\end{aligned} \tag{5.1}$$

where A is the area we aim to partition, and $m = |\mathcal{E}|$ (see Section 2.5 for further explanation). The proposed gradient controller for each agent is then the potential function derivative,

$$u_i = \dot{p_i} = -\frac{d\mathcal{H}_C(p_i)}{\partial p_i} = \alpha\left(-k_p\left(p_i - C_{V_i}\right)\right) + (1-\alpha)\left[-\sum_{i\sim j}\left(\|p_i - p_j\|^2 - d_{ij}^2\right)(p_i - p_j)\right]. \tag{5.2}$$

This proposed controller is a direct weighted combination of two other controllers

with different objectives. When $\alpha = 0$ we have pure formation control, and when $\alpha = 1$ we have pure deployment control. When α is between the values, we may not expect solving both formation and deployment control, and the impact of the coefficient will be discussed later on Subsection 5.2.1. However, there are few interesting results regarding this controller.

Proposition 5.1.1. *Assume that there exists a Central Voronoi Tessellation (CVT), with C_{V_i}, $i = 1, \ldots, l$ such that $\forall i \neq j, \|C_{V_i} - C_{V_j}\|^2 = d_{ij}^2$ (that is C_{V_i} satisfy the formation constraints). Then, there exists an equilibrium $\bar{p}$ for dynamics (5.2) such that $\bar{p} = C_V$ (where $\bar{p}_i = C_{V_i}$ and $C_V = [C_{V_1}, \ldots, C_{V_l}]$), which is locally asymptotically stable.*

Proof. First of all, let us show that $\bar{p} = C_V$ is an equilibrium point of (5.2). Indeed, by assumption, we have $\forall i \neq j, \|C_{V_i} - C_{V_j}\|^2 = d_{ij}^2$, so $\|\bar{p}_i - \bar{p}_j\|^2 = \|C_{V_i} - C_{V_j}\|^2 = d_{ij}^2$, therefore under the assumption we have

$$\begin{aligned}
\dot{p} &= \alpha\left(-k_p\left(\bar{p}_i - C_{V_i}\right)\right) + (1-\alpha)\left[-\sum_{i\sim j}\left(\|\bar{p}_i - \bar{p}_j\|^2 - d_{ij}^2\right)\left(\bar{p}_i - \bar{p}_j\right)\right] \\
&= \alpha\left(-k_p\left(C_{V_i} - C_{V_i}\right)\right) + (1-\alpha)\left[-\sum_{i\sim j}\left(d_{ij}^2 - d_{ij}^2\right)\left(\bar{p}_i - \bar{p}_j\right)\right] \\
&= \alpha\left(-k_p \cdot 0\right) + (1-\alpha)\left[-\sum_{i\sim j} 0 \cdot \left(\bar{p}_i - \bar{p}_j\right)\right] = 0
\end{aligned}$$

showing $\bar{p} = C_V$, is an equilibrium point.

Now we are ready to prove that this equilibrium is stable. Let us choose a candidate Lyapunov function for the combined controller, which is a combination of the Lyapunov functions for each controller ((2.20) for the formation controller and (2.10) for the deployment controller),

$$V_c\left(p\right) = \alpha\left[\sum_{i=1}^{n}\int_{V_i}\min_{i\in\{1,\ldots,n\}}\|q - p_i\|^2\phi(q)dq\right] + (1-\alpha)\left[\frac{1}{4}\sigma(p)^T\sigma(p)\right], \tag{5.3}$$

where $M_{V_i}, J_{V_i}, C_{V_i}$ are as defined in (2.11), (2.12), (2.13) respectively, $p = [p_1, \ldots, p_n]^T$ and $\sigma = [\delta_1, \ldots, \delta_n]^T$ (δ as defined on (2.20)).

As the equilibrium is $\bar{p} = C_V$, we will change the variables such that $\zeta = \bar{p} - C_V$, resulting

$$V_c\left(\zeta\right) = \alpha\left[\sum_{i=1}^{l}\int_{V_i}\min_{i\in\{1,\ldots,l\}}\|q - \zeta_i - C_{V_i}\|^2\phi(q)dq\right] + (1-\alpha)\left[\frac{1}{4}\sigma^T(\zeta + C_V)\sigma(\zeta + C_V)\right]. \tag{5.4}$$

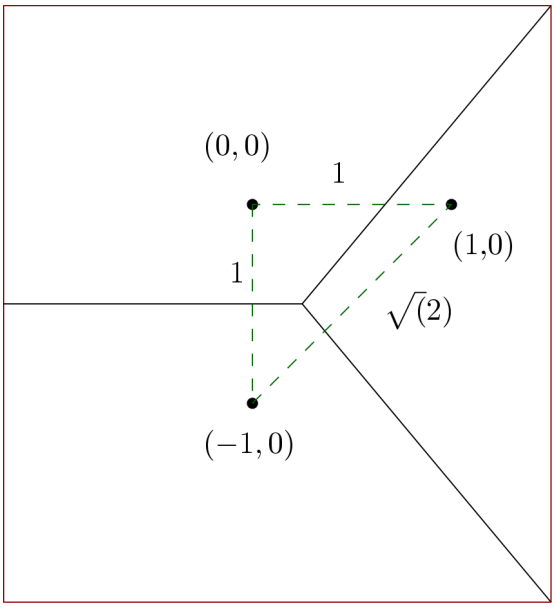


Figure 5.1: An example where deployment equilibrium and formation equilibrium consolidate.

Proof follows the Lyapunov direct method condition as stated on Theorem 2. For condition (1),

$$V_C(\zeta = 0) = \alpha \left[\sum_{i=1}^{l} \int_{V_i} \min_{i \in \{1,\dots,l\}} \|q - C_V\|^2 \phi(q) dq \right] + (1-\alpha) \left[\frac{1}{4} \sigma^T(C_V) \sigma(C_V) \right].$$

We know that $\sigma(C_V) = 0$, as C_V is the equilibrium of the formation controller.

For condition (2), it's obvious that both terms are strictly positive for $\zeta \neq 0$.

For condition (3), the combined derivative of the combined Lyapunov function is [9]:

$$\begin{aligned} \dot{V}_C(\zeta) =& \alpha \left[-2k_p \sum_{i=1}^{n} M_{V_i} \|\zeta\|^2 \right] + \\ & (1-\alpha) \left[-\sigma(\zeta + C_V)^T R(\zeta + C_V) R^T(\zeta + C_V) \sigma(\zeta + C_V) \right] \leq 0. \end{aligned}$$

As we can see, this term is smaller *or equal* to zero. Therefore, using Lyapunov method, this system is stable, however not asymptotically stable. That said, we can notice that the set $S = \{p = C_V\}$ doesn't contain any trajectory apart from the trivial trajectory (when $\zeta = 0$), therefore by LaSalle's principal the system is locally asymptotically stable. ■

An example for Proposition 5.1.1 is as such. Let us assume that we have 3 agents that we want to deploy. The deployment results tessellations, such that the center of mass of the tessellations are $(0,0), (0,1), (0,-1)$. If we have a formation such as in Figure 5.1, then the formation equilibrium is exactly the deployment result.

Proposition 5.1.1 is an example for a case which can be described and analyzed deeply. The general case, however, is highly complicated and non-linear, thus we can't find a general term for the equilibrium. What we can do is analyze its stability.

Theorem 6. *The dynamic system described by the controller (5.2) asymptotically converges to a critical point of the function $G(p) = \alpha\mathcal{H}_{\mathcal{V}}(p) + (1-\alpha)F(p)$, where $F(p)$ is the distance-based formation controller (2.22) potential, and $\mathcal{H}_{\mathcal{V}}(p)$ is the Voronoi-based deployment controller (2.15) potential function.*

Proof. Both the formation control and the deployment controller can be expressed as a gradient dynamics system. It is known that gradient flows converge to the critical points of the potential function. ■

Another major issue is the trade-off between formation and deployment control. Choosing a coefficient different from 1 or 0 means that the final deployment will not be optimal for a deployment nor for formation, thus the engineers of a system that is using that algorithm must know how to handle this. For example, in geolocation missions the geolocating algorithm might suffer heavily from imperfections of the formation, and one solution might be partitioning into more tessellations.

Remark. As the combination of controllers provide some interesting results, it is very important to remember that it doesn't necessarily solve Problem 1. The reason is that the formation is changing the deployment, thus the condition for full area coverage might be broken. This work *does not* provide a condition where the formation control holds *and* Problem 1 is fully solved. Examples are given in Subsection 5.2.1.

5.1.1 Distance-Based Formation Control and PLA

Similar to the regular CVT controller, and as (4.1) is also a stable gradient system, it can also be combined with (2.22), to achieve some spatial properties while maintaining coverage of A_m. As before, first we shall combine the formation control potential function (2.20) with the PLA potential (4.5),

$$\mathcal{H}_{Cp}(p) = \alpha\mathcal{H}_{\mathcal{P}}(P) + (1-\alpha)\,F(p). \tag{5.5}$$

Then, once again, using a coefficient $0 \leq \alpha \leq 1$, the combined controller:

$$u_i = \dot{p_i} = \alpha\left(-k_p\left(p_i - \text{PROJ}_{A_m}\left(C_{V_i}\right)\right)\right) + (1-\alpha)\left[-\sum_{i\sim j}\left(\|p_i - p_j\|^2 - d_{ij}^2\right)(p_i - p_j)\right]. \tag{5.6}$$

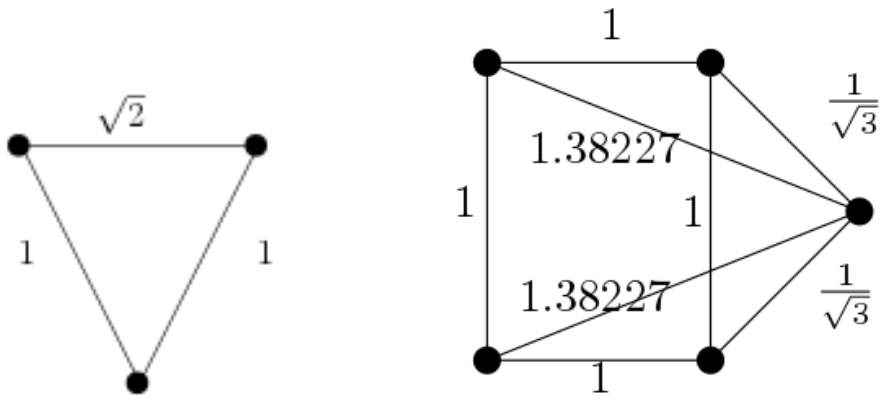
Theorem 7. *The dynamic system described by the controller (5.6) asymptotically converges to a critical point of the function $G(p) = \alpha\mathcal{H}_{\mathcal{P}}(p) + (1-\alpha)F(p)$, where $F(p)$ is the distance-based formation controller (2.22) potential, and $\mathcal{H}_{\mathcal{P}}(p)$ is the same as (4.5).*

Proof. Both the formation control and PLA can be expressed as a gradient dynamical system. It is known that negative gradient flows converge to the critical points of the potential function (Theorem 4). ■

5.2 Simulations and Results

In the following section we present some numerical studies to demonstrate the strategies proposed here. Those analyses are quantifying the errors of different combination coefficients, as well as qualitative analysis of formations and deployment combinations.

5.2.1 Controller Coefficient Analysis



(a) 3 agents formation (b) 5 agents formation

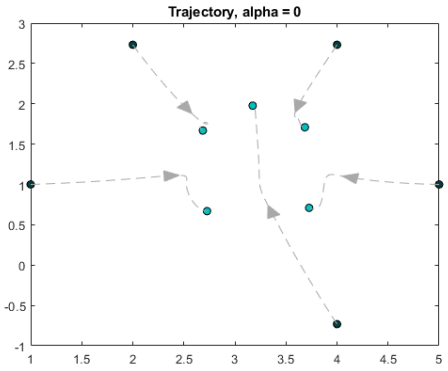
Figure 5.2: Different tested formations for combined coefficient analysis. The numbers above each edge represents the required distances.

In controller (5.2), a convex combination between two controllers was used. In Figure 5.3 we can see the effect of the parameter α on the trajectories of 5 agents, starting in the same initial conditions and trying to achieve formation as described on Figure 5.2(b). The starting position of the agents is represented as black dots while the end of the trajectory marked as teal ones. In this simulation, we aim to explain the significance of α on the trajectory. We do not limit the trajectories (i.e. no physical constraints are presented), however we do define a spatial formation using required distanced between agents. We can clearly see that as α approaches 0, the formation obtains the required shape, and when $\alpha = 1$, the formation convergence to the optimal deployment, in a case of unbounded Voronoi tessellations.

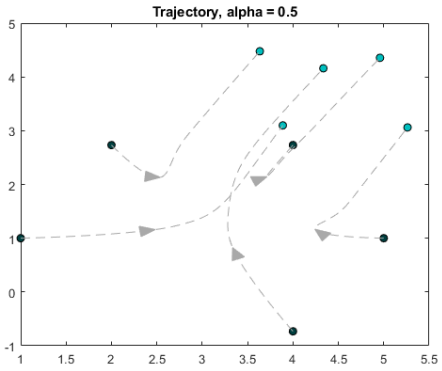
An interesting analysis is checking the influence of α on the formation error δ. For each edge, the edge error δ_k, as introduced in (2.17):

$$\delta_k = \|w_k\|^2 - d_k^2, k \in \{1, \ldots, m\}.$$

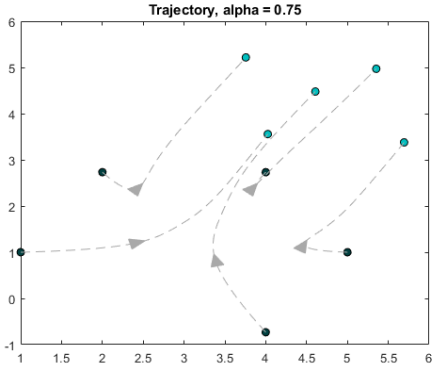
Two different formations were tested, and they can be seen in Figure 5.2.



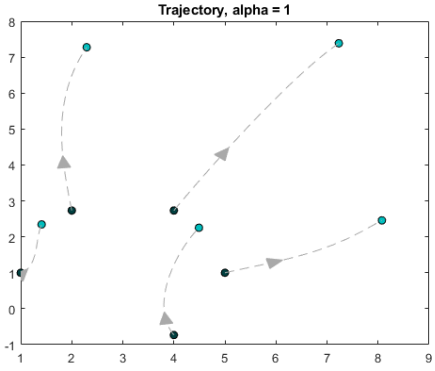
(a) $\alpha = 0$



(b) $\alpha = 0.5$

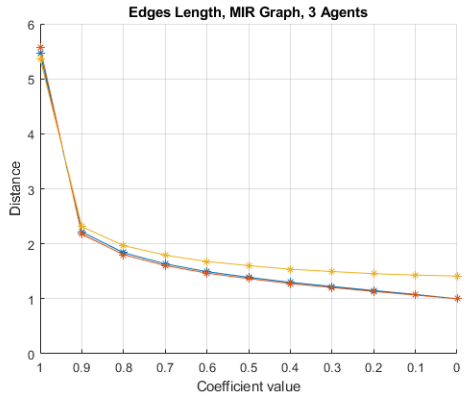


(c) $\alpha = 0.75$

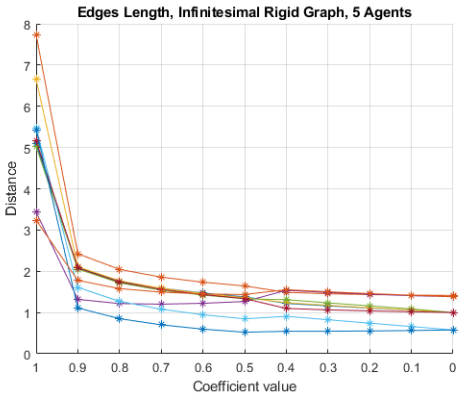


(d) $\alpha = 1$

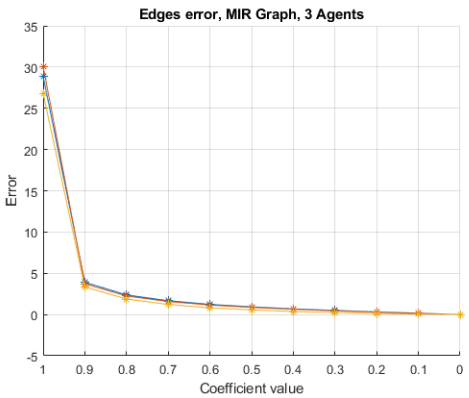
Figure 5.3: Example of trajectories for different α values.



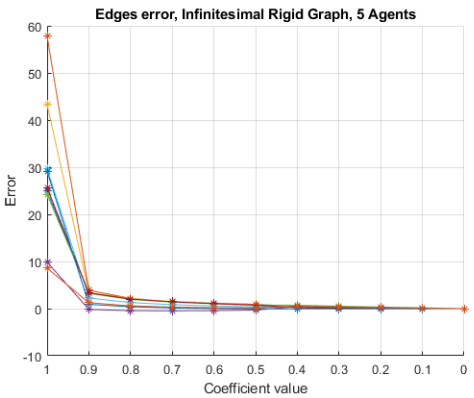
(a) 3 agents formation edges length.



(b) 5 agents formation edges length.



(c) 3 agents formation, edges error.



(d) 5 agents formation, edges error.

Figure 5.4: Edges errors (absolute and relative) for two formations.

We're interested to know what happened in steady-state as a function of α. First we will look into edges length, which can be seen on Figure 5.4(a),(b). It can be clearly seen that the agents are converging into the required formation as the coefficient α approached 0 (which represents a perfect formation). In Figure 5.4(c),(d), we can see the steady-state formation error. In both cases, we can see that the edges error becomes very small (below 1) for $\alpha = 0.3$ and below.

Next, in Figure 5.5, we can see how the total formation error (i.e. the formation potential) behaves. We can see that, as expected, as the coefficient approached 0 the whole formation error approaches zero. In Figure 5.6 we can see the exact same behavior for 5 agents formation.

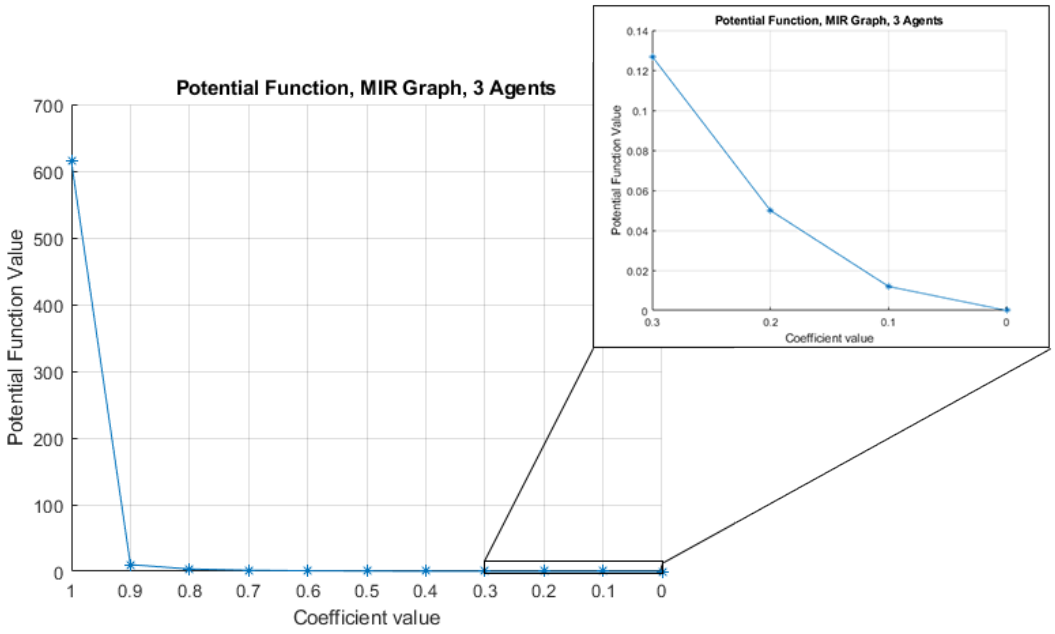


Figure 5.5: 3 agents formation error (potential function value).

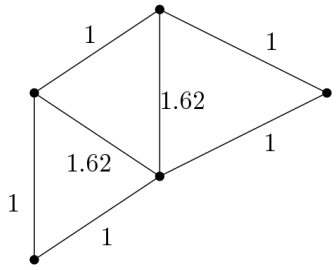


Figure 5.7: Tested formation framework for CVT and formation controller simulation.

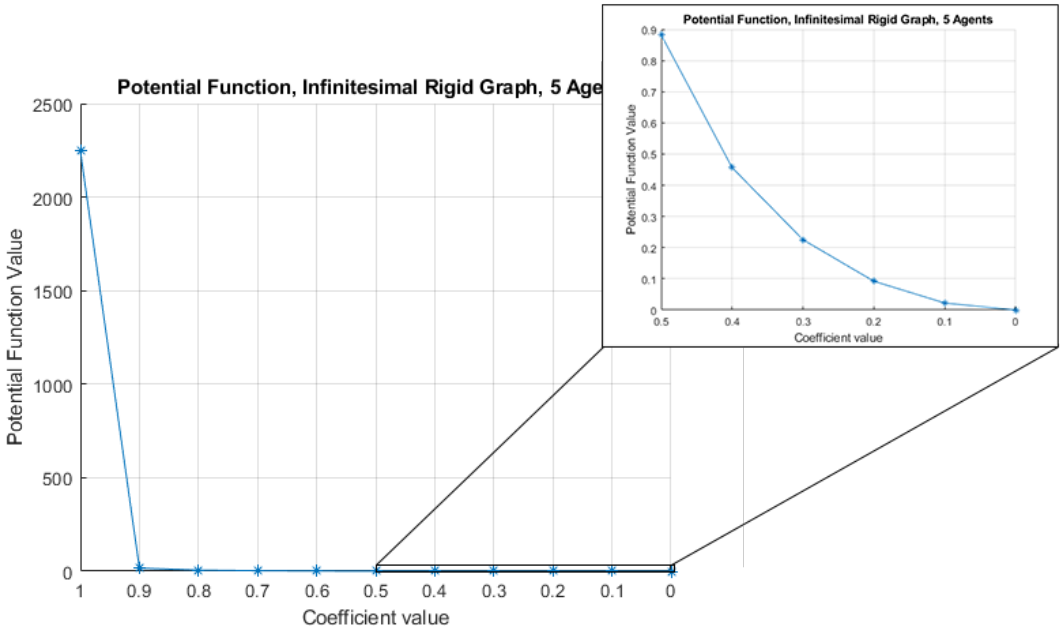
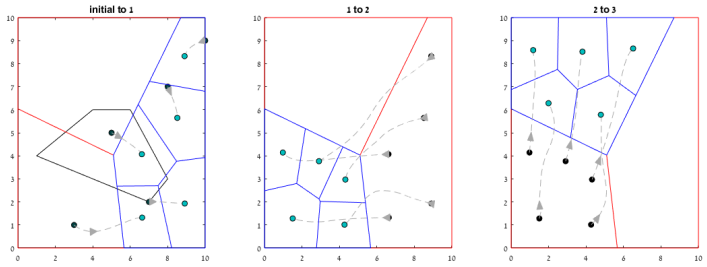


Figure 5.6: 5 agents formation error (potential function value).

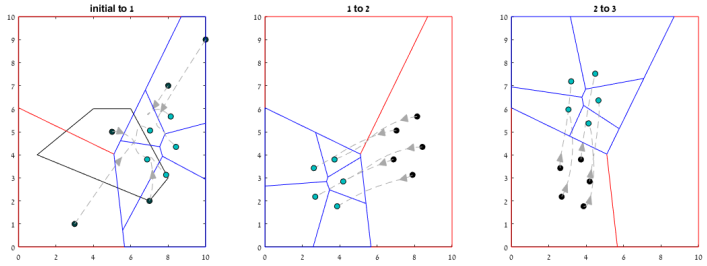
5.2.2 Formation and Deployment Analysis

In the following sub-section, a simulation for Lloyd's algorithm and formation controller (5.2) is presented. In this simulation, we used 5 agents and divided the area A to 3 partitions.

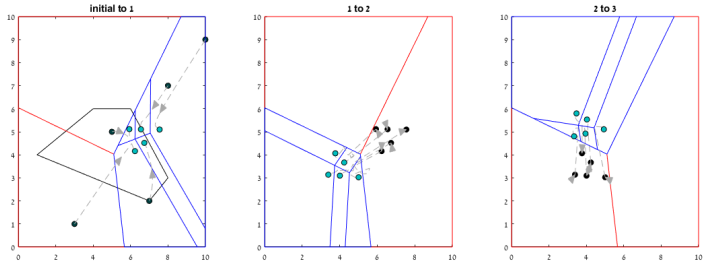
We used Algorithm 4.1 (PLA) to partition the area, and calculated each cell deployment using Lloyd's algorithm (Algorithm 2.1).



(a) $\alpha = 1$ (only deployment).



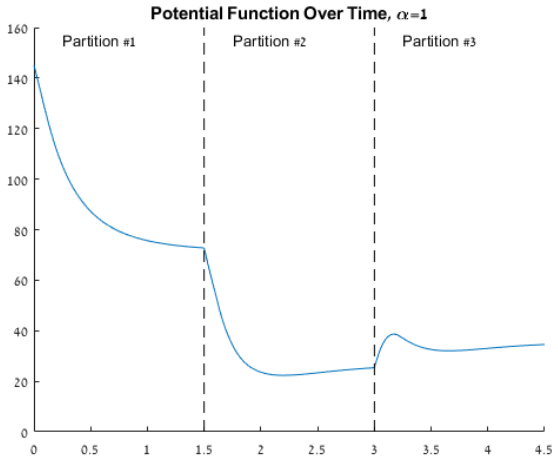
(b) $\alpha = 0.5$ (combination).



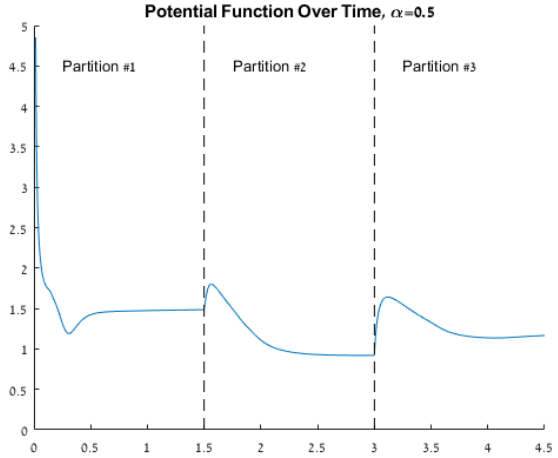
(c) $\alpha = 0$ (only formation).

Figure 5.8: Simulation of formation controller and deployments controller (using Lloyd's algorithm). The subarea A_m is the black polygon, and the active partitions is marked in blue.

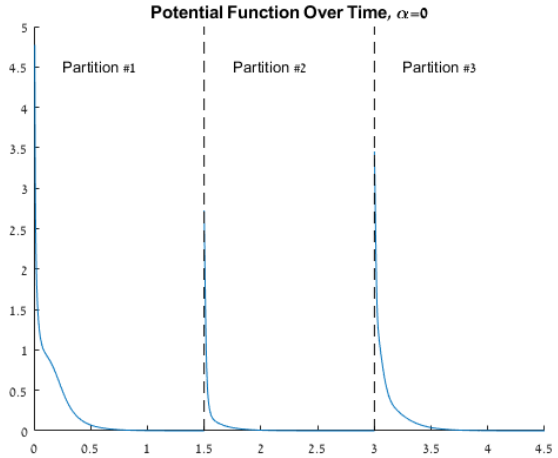
In Figure 5.8, we can see a simulation of 3 cases - deployment only (a), combined controller (b) with $\alpha = 0.5$, and formation only (c). The formation error over time (2.17) varies between the cases, and can be seen on Figure 5.9. In this figure, we see for each cell and each α value how the formation error is changing. We can see that the formation error for $\alpha = 1$ is high as expected, and for $\alpha = 0.5$ we can clearly see how the formation control interfere and reduced dramatically the formation controller, while competing with the deployment control. For $\alpha = 0$ the error is 0 as expected. We can also see clearly for $\alpha = 0, 0.5$ how the potential "grows" when the formation starts to move into a new partition, until convergence within it.



(a) $\alpha = 1$ potential.



(b) $\alpha = 0.5$ potential over time. Values over 5 were cute due to scale reasons.



(c) $\alpha = 0$ potential over time. Values over 5 were cute due to scale reasons

Figure 5.9: Formation potential function value for different cells.

Chapter 6

Conclusion

The problem of partitioning some area aiming to cover it is a well known and researched problem. It was tackled using many approaches - some of them were trajectory-based while the other were partitioning-based. In this work, the goal was covering while maintaining partial coverage on some sub-area and therefore the partition-based strategy was chosen.

In many works (for example, [9]), the partitioning optimization was based on Centroidal Voronoi Tessellations (CVT), which provides a reasonable strategy for partitioning an area when the goal is covering it. The most popular way for calculating the CVT was using an algorithm known as Lloyd's Algorithm [14, 9]. In this work, the Projected Lloyd's Algorithm (PLA) was introduced. This algorithm utilized a projection operator for partitioning an area while still covering a constraining sub-area.

The PLA resulted partitioning that intersects with the constraining sub-area, creating interesting partitions which varies with initial conditions - and this can be utilized for practical usages.

During the work, another question was raised - what happens if we want to maintain some formation, creating some desired spatial structure. The results can be beneficial for many practical use cases. In this work the most obvious example was geolocation, but of course - there might be many more usages. This question was answered, using a combination of deployment and formation controllers, and specifically a distance-based formation controller. The combination was tested mostly for the trade-off between the two controllers, and the results shows that although it's highly depended on the framework, starting from $\alpha = 0.3$, the formation is generally well controlled. Further analysis is required in future work.

6.1 Future Work and open questions

First of all, as the proposed algorithm for PLA is centralized, the first and most obvious question that comes to mind is what happens if there isn't any central computer available. Therefore, a *decentralized method* should be developed.

Secondly, in Chapter 5, while the two controllers where combined using a coefficient, we did not supply any *condition to ensure some minimal coverage or maximal formation error*. Although simulation supplied some intuition regarding the formation error, this should be analyzed more formally.

Third, we could not find the combined Lloyd's Algorithm and Distance-based formation controller *critical points*.

Lastly, one major assumption was made in this work - that no matter what the PLA results are, the agents can cover it in one static formation. This assumption obviously does not hold in real-world problem, therefore an algorithm for partitioning the area using PLA *and* ensuring coverage for each partition is mandatory. This algorithm can utilize the results for the formation and deployment combination coefficient.

Bibliography

[1] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of multiple UAVs for persistent surveillance: Algorithm and flight test results," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1236–1251, 2012.

[2] J. M. Palacios-Gasós, E. Montijano, C. Sagüés, and S. Llorente, "Multi-robot persistent coverage with optimal times," in *55th IEEE Conference on Decision and Control*, pp. 3511–3517, Dec 2016.

[3] S. G. Loizou and C. C. Constantinou, "Multi-robot coverage on dendritic topologies under communication constraints," in *55th IEEE Conference on Decision and Control*, pp. 43–48, Dec 2016.

[4] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. 2010.

[5] C. G. Cassandras and Wei Li, "Sensor networks and cooperative control," in *44th IEEE Conference on Decision and Control*, pp. 4237–4238, Dec 2005.

[6] G. M. Atınç, D. M. Stipanović, P. G. Voulgaris, and M. Karkoub, "Supervised coverage control with guaranteed collision avoidance and proximity maintenance," in *52nd IEEE Conference on Decision and Control*, pp. 3463–3468, Dec 2013.

[7] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 859–865, 1991.

[8] S. Rutishauser, N. Correll, and A. Martinoli, "Collaborative coverage using a swarm of networked miniature robots," *Robotics and Autonomous Systems*, vol. 57, pp. 517–525, may 2009.

[9] J. Cortes and S. Martinez, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[10] I. I. Hussein and D. M. Stipanovic, "Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[11] S. Shakkottai, R. Srikant, and N. Shroff, "Unreliable sensor grids: coverage, connectivity and diameter," in *22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1073–1083 vol.2, March 2003.

[12] Wei Li and C. G. Cassandras, "Distributed cooperative coverage control of sensor networks," in *44th IEEE Conference on Decision and Control*, pp. 2542–2547, Dec 2005.

[13] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[14] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[15] K. K. Oh, M. C. Park, and H. S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[16] M. Basiri, A. N. Bishop, and P. Jensfelt, "Distributed control of triangular sensor formations with angle-only constraints," in *ISSNIP 2009 - Proceedings of 2009 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, vol. 59, pp. 121–126, Elsevier B.V., 2009.

[17] X. Sun and C. G. Cassandras, "Optimal dynamic formation control of multi-agent systems in constrained environments," *Automatica*, vol. 73, pp. 169 – 179, 2016.

[18] F. Mohseni, A. Doustmohammadi, and M. B. Menhaj, "Distributed receding horizon coverage control for multiple mobile robots," *IEEE Systems Journal*, vol. 10, pp. 198–207, March 2016.

[19] M. Schwager, D. Rus, and J. J. Slotine, "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," *International Journal of Robotics Research*, vol. 30, no. 3, pp. 371–383, 2011.

[20] C. Godsil and G. Royle, *Algebraic Graph Theory*, vol. 207 of *Graduate Texts in Mathematics*. New York, NY: Springer New York, 2001.

[21] L. Asimow and B. Roth, "The rigidity of graphs, II," *Journal of Mathematical Analysis and Applications*, vol. 68, no. 1, pp. 171–190, 1979.

[22] R. Connelly, "Generic Global Rigidity," *Discrete & Computational Geometry*, vol. 33, pp. 549–563, apr 2005.

[23] D. J. Jacobs and B. Hendrickson, "An Algorithm for Two-Dimensional Rigidity Percolation: The Pebble Game," *Journal of Computational Physics*, vol. 137, no. 2, pp. 346–365, 1997.

[24] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, R. Yang, B. D. Anderson, and P. N. Belhumeur, "A theory of network localization," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1663–1677, 2006.

[25] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilization of infinitesimally rigid formations of multi-robot networks," in *47th IEEE Conference on Decision and Control*, pp. 477–482, Dec 2008.

[26] B. Roth, "The Rigidity," *Journal of Mathematical Analysis and Applications*, 1979.

[27] S. Zhao and D. Zelazo, "Bearing Rigidity Theory and Its Applications for Control and Estimation of Network Systems: Life Beyond Distance Rigidity," *IEEE Control Systems Magazine*, vol. 39, no. 2, pp. 66–83, 2019.

[28] T. S. Tay and W. Whiteley, "Recent Advances in the Generic Ridigity of Structures," *Topologie structurale*, vol. 9, pp. 31–38, 1984.

[29] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002.

[30] S. Sastry, *Nonlinear Systems Analysis, Stability, and Control*. Interdisciplinary Applied Mathematics, Springer New York, 1999.

[31] G. R. Sell, "Stability theory and Lyapunov's second method," *Archive for Rational Mechanics and Analysis*, vol. 14, no. 1, pp. 108–126, 1963.

[32] F. Bullo, *Lectures on Network Systems*. CreateSpace, 1 ed., 2018. With contributions by J. Cortes, F. Dorfler, and S. Martinez.

[33] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations," *SIAM Journal on Numerical Analysis*, vol. 44, pp. 102–119, jan 2006.

[34] Kwang-Kyo Oh and Hyo-Sung Ahn, "Distance-based formation control using Euclidean distance dynamics matrix: General cases," *2011 American Control Conference*, pp. 4816–4821, 2011.

[35] X. Guo, Y. Zhang, and B. Zeng, "Passive Localization Using Time Difference of Arrival and Frequency Difference of Arrival," *Journal of Computer and Communications*, vol. 06, no. 01, pp. 65–73, 2017.

על מנת להתמודד עם אילוץ 1 ו-3, נדרשנו לתת מענה ייחודי. בעזרת הצעה לאופרטור הטלה (projection operator), הוצע "אלגוריתם לויד המוטל" (Projected Lloyd's Algorithm). זהו אלגוריתם המשנה מעט את אלגוריתם לויד המקורי, כך שהוא מטיל בכל איטרציית חישוב את התוצאה אל השטח המאלץ בעזרת אותו אופרטור הטלה. בעזרת השימוש באלגוריתם זה, הצלחנו להבטיח התכנסות אל מחיצות אשר לכל הפחות נחתכות עם השטח המוטל. האלגוריתם אופיין ואף נבנה בקר פריסה מתאים, והוכח שהבקר מתכנס אסימפטוטית מקומית אם כי קיים קושי לאפיין את נקודות שיווי המשקל שלו עקב חוסר הלינאריות של הבעיה. אלגוריתם הפתרון הכולל, מתחיל בחלוקת השטח למחיצות על ידי אלגוריתם לויד המוטל, ולאחר מכן משתמש ב"מחיצות וורונוי ממורכזות" עבור כיסוי בכל מחיצה אשר נוצרה על ידי האלגוריתם המוטל.

לאחר שניתן מענה לבעיה זו, פנינו לעסוק בבעיה נוספת. לעיתים יש משמעות מאוד גדולה למבנה המרחבי של החיישנים בעת הפריסה. דוגמה קלאסית (אך לא יחידה כמובן) היא בעיית האיכון – שם למבנה המרחבי של החיישנים השפעה דרמטית על איכות האיכון. לכן, נבחר אלגוריתם בקרת מבנה (Formation control) מסוג שמירת מרחק (Distance-based) בו אנחנו מעוניים לשמור על המרחק בין חיישנים אשר מסוגלים לדבר אחד עם השני. בקר המבנה שולב עם בקרי פריסה – הן עם אלגוריתם לויד והן עם אלגוריתם לויד המוטל, אם כי ליישום הזה אלגוריתם לויד מתאים יותר, והבקר הוכח כיציב אסימפטוטית מקומית. הוצגו מספר סימולציות והשוואות. חיבור הבקרים בוצע עם מקדם ונבחנה השפעת המקדם על המבנה. עם זאת, לא הוצגו תנאים אשר מגדירים סף מסוים למבנה או סף מסוים לאיכות הכיסוי.

לסיכום, בעבודה זו פותחו שני בקרים – הראשון מחשב מחיצות עם אילוצי חיתוך מחיצה בפוליגון, והשני בקר המשלב בין בקרת מבנה לבין בקרת פריסה. הבקרים הוכחו כיציבים והוצגו אלגוריתמים וסימולציות המדגימים שימוש בבקרים אלו.

תקציר

עבודה זו עוסקת בבקרת כיסוי דינמית עבור שטחים גדולים בעזרת סוכנים (חיישנים) ניידים, כאשר לא קיימים מספיק חיישנים לקבל כיסוי מלא.

בעיית כיסוי שטח היא בעיה ידועה אשר נחקרה ועודנה נחקרת תחת אילוצים שונים. הצורך במענה לבעיה ברור ויש לו שימושים רבים – מעקב אחרי מטרות, איכון אובייקטים (בתהליך הפוך לאיכון לווייני), חיפוש קרקעי ואפילו שואבי אבק רובוטיים אשר הולכים ונהיים פופולריים בשנים האחרונות.

אחד התחומים הנחקרים הוא בעיית הכיסוי החלקי – כאשר לא קיימים מספיק חיישנים לכיסוי מלא של השטח. במקרה זה, קיימות שתי אסטרטגיות עיקריות למנגנון כיסוי השטח של החיישנים. הראשונה – אסטרטגיה בה מוגדר מסלול לכל חיישן, כך שסך השטח המכוסה על ידי כל החיישנים לאורך כל המסלולים יהווה כיסוי מלא של השטח. מימוש של אסטרטגייה זו ניתן לראות, למשל, על ידי שואבי אבק רובוטיים אשר סורקים את החדר תוך כדי תנועה עד להשלמת המשימה (גם אם במקרה זה מדובר בחיישן יחיד). אסטרטגייה שניה היא בעזרת חלוקת השטח למחיצות, כך שכל מחיצה ניתנת לכיסוי על ידי סט החיישנים על ידי אלגוריתם פריסה כלשהו. מעבר בין כלל המחיצות יבטיח כיסוי מלא של השטח. בעבודה זו בחרנו להתמקד בבעית הכיסוי תחת מספר הנחות ואילוצים:

1. לא קיימים מספיק חיישנים לכיסוי מלא של השטח,
2. המשתמש ירצה לשמור כיסוי לעיתים על שטח מסוים,
3. המשתמש ירצה לשמור כיסוי חלקי לכל הפחות לשטח נתון מראש.

לאור ההנחות והאילוצים, בחרנו להשתמש באסטרטגיה בה השטח מחולק למחיצות.

בספרות, השיטה הנפוצה ביותר היא באמצעות חלוקה ל"מחיצות וורונוי ממורכזות" (Centroidal Voronoi Tessellations). "מחיצות וורונוי" היא שיטה לחלוקת שטח באופן הבא – בהינתן שטח ואוסף מחוללים עליו (אשר יכולים לייצג את החיישנים, לדוגמה), המחיצה המתאימה למחולל מסויים היא אוסף כלל הנקודות בשטח הקרובות ביותר לאותו מחולל. כאשר אותו מחולל מהווה את מרכז המסה של אותה מחיצה – זוהי "מחיצת וורונוי ממורכזת". השיטה בה מחלקים את השטח ל-"מחיצות וורונוי ממורכזות" היא שיטה ידועה לחלוקה אופטימלית של חיישנים על פני שטח ובהינתן שרדיוס הכיסוי של החיישנים גדול מספיק – מהווה פתרון האופטימלי לבעיה. החישוב מתבצע בעזרת אלגוריתמים שונים, הנפוץ הינו אלגוריתם לויד (Lloyd's Algorithm) מובן שמדובר בחישוב מאוד מורכב והאופטימום הינו מקומי, אך הבקר יציב אסימפטוטית מקומית!

המחקר בוצע בהנחייתו של פרופסור דניאל זלזו, בפקולטה להנדסת אוירונאוטיקה וחלל.

חלק מן התוצאות בחיבור זה פורסמו כמאמרים מאת המחבר ושותפיו למחקר בכנסים ובכתבי-עת במהלך תקופת המחקר של המחבר, אשר גרסאותיהם העדכניות ביותר הינן:

Yoav Palti and Daniel Zelazo. A Projected Lloyd ' s Algorithm for Coverage Control Problems. In *Proceedings of The 59th Israel Annual Conference on Aerospace Sciences*, pages 1008–1022, Tel Aviv & Haifa, march 2019.

תודות

אני רוצה להודות לפרופ"ח דניאל זלזו על תמיכתו במהלך עבודתי על תזה זו, על שנתן לי להביע את עצמי ואת רעיונותיי. ארצה להודות למפא"ת על תמיכתם החלקית במחקר זה.

אסטרטגיות פריסה עבור בעיות בקרת כיסוי

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת אוירונאוטיקה וחלל

יואב פלטי

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
תמוז התשע"ט חיפה יולי 2019

אסטרטגיות פריסה עבור בעיות בקרת כיסוי

יואב פלטי