DISTRIBUTED AREA PARTITIONING FOR MULTI-ROBOT COVERAGE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BURAK HOCAOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2022

Approval of the thesis:

DISTRIBUTED AREA PARTITIONING FOR MULTI-ROBOT COVERAGE

submitted by **BURAK HOCAOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Erol Şahin
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Veysel Gazi
Control & Automation Engineering, Yıldız Technical University _____

Assoc. Prof. Dr. Erol ŞAHİN
Computer Engineering, METU _____

Prof. Dr. Faruk Polat
Computer Engineering, METU _____

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Engineering, METU _____

Assist. Prof. Dr. Hakan Yıldız
Computer Engineering, METU _____

Date: 09.05.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Burak Hocaoğlu

Signature :

ABSTRACT

DISTRIBUTED AREA PARTITIONING FOR MULTI-ROBOT COVERAGE

Hocaoğlu, Burak
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Erol Şahin

May 2022, 48 pages

This thesis focuses on the distributed coverage area partitioning problem in a robotic swarm deployed over a known, two-dimensional environment. The goal of the coverage area partitioning problem is to split a region of interest into a number of subregions such that the amount of effort required to cover each subregion is approximately equal with respect to an objective function. Towards this end, the problem is defined as a distributed optimization problem, where the decision varibles are the locations of robots. Then, the proposed solution approach is described as a control mechanism driving each robot to better positions. At the end, robot positions will converge to a configuration where the total effort to process their subregion is minimal. Specifically, first, *Lloyd's Algorithm*, [1], is deployed on the base approach to workload partitioning by converging to a *Centroidal Voronoi Tesselation (CVT)*. Lloyd's Algorithm provides a basic idea for convex problems where the deployment region has convex boundaries, but fails to provide reliability when used in a non-convex region. At this point, heuristics are added to reinforce the capabilities of Lloyd's Algorithm to capture the dynamics missed when using CVT and to enable stable convergence in non-convex problems, converting the behaviour to converging to a *Geodesic Voronoi Tesselation (GVT)*. Also, the model as the objective function and the control mecha-

nism is revised for equitable workload partititoning. After that, various analyses are conducted with different centrality metrics whose results give the actual target locations that agents need to arrive, while heuristics refine how each agent arrives. These centrality metrics add new dynamics to how local workloads of each agent is defined, as well as optimizing them. This work provides qualitative and quantitative results of the proposed approaches. The experiments are done in non-convex two-dimensional environments where Lloyd's Algorithm fails in its base form.

Keywords: swarm robotics, distributed area partitioning, centroidal Voronoi tessellation, distributed coverage control

ÖZ

ÇOKLU-ROBOT TARAMADA DAĞITIK ALAN PAYLAŞIMI

Hocaoğlu, Burak
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Erol Şahin

Mayıs 2022 , 48 sayfa

Bu tez çalışması bilinen iki boyutlu bir ortamda dağıtılan sürü robotlarında dağıtık alan paylaşımı problemine odaklanmaktadır. Tarama alanı paylaşımı problemi, ilgili bir bölgenin belirli bir sayıda parçalara ayrılmasını ve bu parçaların bir görev tanımına ya da bedel fonksiyonuna göre işlemenin minimum eforla gerçekleşmesini ele alır. Bu doğrultuda, problem, ajan pozisyonlarına bağlı olan bir dağıtık optimizasyon problemi formatında tanımlanacaktır. Ana hedef genel iş yükünü eşit miktarda paylaşmaktır. Genel iş yükü, her bir ajanın bölgedeki diğer noktalara olan uzaklığıyla doğru orantılıdır. Aynı zamanda da, bu ajanları paylaşımın optimum olduğu pozisyonlara doğru bir rotada yönlendiren yerel bir kontrol mekanizmasının yardımıyla paylaşılır. Ajan pozisyonları olan karar değişkenleri de bu konfigürasyona ulaştığı zaman ilgili bölgeyi işlemek için harcanacak toplam eforun minimum olduğu bir şekilde parçalar. Spesifik olarak *Lloyd Algoritması*, [1], kullanılarak temel yaklaşımın *Merkezi Voronoi Teselasyonu (CVT)* olduğu durum incelenecektir. Lloyd Algoritması ajanların dağıtıldığı konveks dış sınırları olan bölgelerde basit bir fikir sağlamaktadır, fakat genel bir konkav problem karşısında güvenilirlik sağlayamaz. Bu aşamada CVT kullanırken hesaba katılmayan dinamikleri yakalamak ve konkav problemlerde stabil yakınsama

sağlamak için Lloyd Algoritması'na buluşsal yöntemler eklenecektir ve Lloyd Algoritması'nın özündeki yakınsama davraınşı *Jeodezik Voronoi Teselasyonu'na (GVT)* benzeyecektir. Görev fonkiyonu ve kontrol mekanizması da iş yükünün eşit paylaşımı için güncellenecektir. Daha sonra, ajanların, buluşsal yöntemlerin gidişatı ayarladığı, varmaları gereken ara hedefleri veren farklı merkezilik metrikleri ile analizler yapılacaktır. Bu merkezlik metrikleri ajanların lokal iş yükünün hesabında ve bunların optimizasyonunda yeni dinamikler ekleyecektir. Bu çalışma sunulan yaklaşımların kalitatif ve kantitatif sonuçlarını sağlamaktadır. Deneyler, Lloyd Algoritmasının temel halinde başarısız olduğu konkav iki boyutlu alanlarda yapılmıştır.

Anahtar Kelimeler: robot sürüleri, dağıtık alan paylaşımı, merkezi Voronoi teselasyonu, dağıtık tarama kontrolü

To everyone who helped me to get here...

ACKNOWLEDGMENTS

First of all, I'd like to thank my advisors Erol Şahin and Hakan Yıldız for their tolerance especially with my "depth-first thinking" that led me into so many blind alleys throughout my studies. Their acceptance of my professional and academic ambitions, their guidance, advice and criticism helped me to see the bigger picture.

I'd also thank my teachers Sinan Kalkan, Ali Emre Turgut for their contributions to my academic and professional responsiblities, allowing me to see and to learn different perspectives, but especially shaping the way I think about conducting research and how I can carry myself as an academician. I was, am and will always be happy to seek their guidance and enlighten ourselves in doing what we do.

I'd like to thank my family Hilmi Hocaoğlu, Fatma Hocaoğlu and Rıdvan Berkay Hocaoğlu for putting up with my bickering, my headstrong and thick-headed behaviour, my God-knows-how-many complaints about many things, allowing me and supporting me with all of my goals, ambitions and life decisions, no matter how the end results were and may be.

I'd like to thank my friends and colleagues from various places that I worked for, Tugay Alperen Karagüzel, İlyas Eren Yılmaz, Emre Gür, Cihangir Ümithan Göktolga, Göktuğ Hamdi Görgel, Cem Bilaloğlu, Umutcan Üstündaş, Giray Önür, Samet Elmacı, Şamil Dinçer, Arash Sadeghi, Fatih Can Kurnaz, Erfan Khalaji, Mert Kaan Korkmaz, İdil Sülo, Yunus Terzioğlu, who helped me build the foundations of my thinking, any study that we or I conducted, both in an academic perspective and/or professional perspective, or just being there doing what was to be done.

Finally, I'd like to thank my friends Tahsincan Köse, Raşit Özdemir, Cansın Canberi and Güneş Sucu; Barış Bayraktar, Yunus Demirören, Cihan Akdeniz, Kaan Furkan Altınok for their sense of humor, fun and honest approach to anything we may have encountered in our professional, academic and daily lives together or as individuals sharing among ourselves.

Should someone read this and get upset because I forgot them? My sincere apologies and thank you very much for your touch in my life.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xii

LIST OF TABLES xv

LIST OF FIGURES xvi

LIST OF ABBREVIATIONS xix

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Main Challenge 1

1.3 The Outline of the Thesis 2

2 BACKGROUND KNOWLEDGE 3

2.1 Coverage Area Partitioning 3

2.1.1 Placement vs. Subregion 4

2.1.2 Voronoi Tessellation 6

2.2 Lloyd's Algorithm in Coverage Control 6

2.3 Geometric Properties of Coverage Region 7

2.4 Distance Metric in Lloyd's Algorithm 9
2.5 Geodesic Distance Metric in Lloyd's Algorithm 11
2.5.1 Locating the Geodesic Centroid 11
2.5.2 Discretization of Coverage Area Partitioning 13
2.5.2.1 Straight Skeleton of a Polygon 13
2.5.2.2 Metric Grid of a Polygon 14
2.5.3 Gradient of the Geodesic Lloyd's Algorithm 15
3 RELATED STUDIES 19
3.1 Voronoi Tessellation as Area Partitioning 19
3.1.1 Locate-and-Move-to-Centroid 20
3.2 Geodesic Distance Metric 20
3.3 Equitable Partitioning 20
3.4 Geometric Re-interpretations 21
3.4.1 Straight Skeleton Method 21
3.4.2 Grid & Graph Methods 22
4 DISTRIBUTED AREA PARTITIONING FOR COVERAGE 25
4.1 Our Methodology 26
4.1.1 Subregion Computation Stage 26
4.1.2 Roadmap Generation Stage 27
4.1.3 Gradient Control Stage 27
4.2 Method 1: ED + SS + Naive Control 27
4.2.1 Performance 30
4.3 Method 2: GD + MG + Equitable Control 33

4.3.1 Performance 35

4.4 General Comparison 37

5 CONCLUSIONS 43

5.1 Future Work 44

REFERENCES 45

LIST OF TABLES

TABLES

Table 4.1 Individual workload standard deviation scores of Method 1 in 6 different scenarios. The values for Spiral region are obtained at a coverage percentage less than 100%, as Table 4.2 shows. 30

Table 4.2 Coverage percentage scores of Method 1 in 6 scenarios ($\%/100$). Overlaps are allowed to some extent to ease the precision arithmetic within CGAL; hence, the $\mathbf{1.019}$ value. 34

Table 4.3 Workload standard deviation scores of Method 2 in 6 different scenarios. Left group is based on the number of navigable locations, while right group is area based. 40

Table 4.4 Coverage percentage scores of Method 2 in 6 scenarios ($\%/100$). . . 40

LIST OF FIGURES

FIGURES

Figure 2.1 Two placement schema as a visualization to base station example. Each nested circular groups represent a coverage for the base station placed in the center of them. The coverage degrades as the distance to the base station increases, so the closer to any center, the better. On the left, some sections of the region gets poor service by the stations due to long distances. On the right, however, a better coverage quality over the entire region can be achieved with a more *encompassing* placement. Note that, each section of the region has a lower maximum distance to an assigned station on the right compared to that on the left. 5

Figure 2.2 Convergence to a CVT with 9 agents each running Lloyd's Algoithm in 2D polygonal regions. The circles and their tracing lines represent the final robot locations and their trajectories, respectively. . . 8

Figure 2.3 The gradient control in the base Lloyd's Algorithm would fail in non-convex environments. The trajectories following the centroids can go through unnavigable sections like holes (obstacles) and boundaries or the partitioning can converge to unnavigable locations. 10

Figure 2.4 Two VTs with Euclidean distance metric are applied on a non-convex region. The circles indicate the robot positions. As the robots move closer to each other, there comes an instant, when the bisector starts to intersect with the boundary of the region more than twice. When this happens, disconnected subregions will appear. 11

Figure 2.5 Applying Geodesic distance metric to Figure 2.4. In (a), the upper section is a more logical partitioning, since the blue robot has to do extra movements to reach the boundary due to the non-convexity. Therefore, the bisector curves into the blue subregion indicating equal arrival effort rather than straight distance. In (b), Geodesic bisector reduces down to a regular Euclidean bisector; however, the disconnected piece in Figure 2.4b is now attached to the red robot, since it's shortest path to that piece is shorter than that of blue robot's. 12

Figure 2.6 The red line segments are the edges of the graphs connecting the boundary vertices to the inner vertices. These segments are also the bisectors partitioning the polygon into subregions that display the geometry of the polygon. 14

Figure 2.7 Straight Skeleton of a rectangle. If the aspect ratio decreases, the geometry would look more like a square and the center of the Straight Skeleton can approximate the actual centroid (blue dot). 15

Figure 2.8 The small green circles represent navigable locations. Note that instead of a sparse navigation graph, we obtain a dense navigation mesh. 16

Figure 2.9 (a) Overlapping boundaries mean that one robot's movement can affect the other's partitioning. (b) The move-to-centroid behaviour (in black) of the standard Lloyd's Algorithm. (c) Additional forces (in gray), due to boundary sharing in (a), to move the robots to placements, where the partitioning is more equivalent. The $\mathbf{m}_{i,j}$ are middle points for each neighbouring pair. 18

Figure 4.1 Scenarios for CAP experiments 25

Figure 4.2 Method 1 performance in minimization of individual workload standard deviation over iterations 31

Figure 4.3 Navigation traces in Corridors region: Standard Lloyd's Algorithm vs. Method 1. Blue dots indicate the convergence locations over all 10 runs. 32

Figure 4.4 Method 1 is not safe from the bias due to the inital placement. Combined with ED in stage one, disconnectivities may happen at any iteration. We do not assign the disconnected subregions for a more logical and practical partitioning. 33

Figure 4.5 A possible GVP with 5 agents using 4-connectivity. 37

Figure 4.6 Method 2 performance in minimization of workload standard deviation over iterations. The red plots are the standard deviation of the number of navigable locations assigned to each robot, whereas the blue plots are that of the area calculated with $resolution^2 * count$. 38

Figure 4.7 General comparsion of the algorithms with 4-agent and 8-agent cases over the regions 39

Figure 4.8 Possible partitioning of Corridors region with 4 agents using Method 2. Blue robot occupies most of the shared section. In such cases, the dominance tuning of $\mathbf{u}_i^{workload}$ is poor. 41

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
CVT	Centroidal Voronoi Tessellation
GVT	Geodesic Voronoi Tessellation
CAP	Coverage Area Partitioning
PD	Power Diagram
VT	Voronoi Tessellation
FW	Floyd-Warshall
BCD	Boustrophedon Cell Decomposition
PRM*	Probabilistic Roadmap-Star
BFS	Breadth-First Search

CHAPTER 1

INTRODUCTION

1.1 Motivation

Recent advances in robotics enabled the deployment of multi-robot systems in remote and dangerous surveillance and search-and-rescue tasks that cover large areas. For example, in aerial surveillance missions, the use of drone swarms decrease the cost and the risks associated with the use of humans for the same task. The time-of-completion is of utmost importance, and hence it is essential to partition the area into equal disjoint subregions that can be covered by individual robots. By partitioning, the time-of-completion is expected to decrease due to the parallelism introduced by the multi-agent setup.

1.2 Main Challenge

In multi-agent setups, the main challenge is to have a solution that can be executed in a distributed fashion to make the best use of multiple processing units (i.e. agents). Moreover, in order to make the most efficient use of it, this distributed solution should aim for equivalent division of the total workload.

Considering the mobility of robots, partitioning is usually driven by a control algorithm moving each robot in the directions resulting in the desired outcome. Finding the right sequence of targets towards which the robots will move to improve the overall state is not trivial, as we will find out later. Instead of computing the exact solution offline, approximating it with online computations are preferred in practice; therefore, we chose an iterative optimization approach.

Another challenge is introduced by the region of interest to be partitioned. In case of obstacles inside the region and/or the region has a non-convex shape, the iterative optimization algorithm should take the geometric constraints into consideration when computing the next target. As we will see later, such factors can dramatically increase the complexity in certain steps of computation.

1.3 The Outline of the Thesis

Chapter 2 introduces the main concepts. Next, in Chapter 3, we discuss how our technical background is used in previous studies. Then, in Chapter 4, we propose our algorithms for the experiments. The implementations of the heuristics introduced in Chapter 2 will be shown dealing with the shortcomings of the standard approach. We display performances in three scenarios and a baseline scenario where the standard approach gives a credible solution. Also, we share our findings in what factors influenced our results and where our algorithms lack providing better solutions. Finally, in Chapter 5, the study will be summarized and the general framework will be reviewed for the future work.

CHAPTER 2

BACKGROUND KNOWLEDGE

This chapter introduces the main concepts and definitions about the *Coverage Area Partitioning (CAP)* problem. In Section 2.1, the CAP problem and the solution intuition are defined. Then, Section 2.2 introduces the Lloyd's Algorithm, which forms the basis of our approach to evaluate and improve for the CAP problem. Section 2.3 discusses the geometric properties of deployment regions and their influence to the solution of CAP and the issues of Lloyd's Algorithm subject to these properties. Section 2.4 reviews the distance metrics and their influence to the solution of CAP. Finally, Section 2.5 describes a version of Lloyd's Algorithm supported by the Geodesic distance metric, which is used in our experiments. In addition, we show two ways how the CAP problem can be discretized (2.5.2) to avoid practically intractable computation due to what the Geodesic distance metric requires.

2.1 Coverage Area Partitioning

The Coverage Area Partitioning problem involves the splitting of an area into a set of disjoint subregions. The goal is to find a set of subregions such that the cost of sensing the entire region is minimized. Here, the cost is defined as the amount of local work each agent has to do to cover its subregion. Throughout this thesis, *coverage* is used as a placeholder term to indicate the information collection from the whole region at hand via the sensors on-board the robot. For instance, coverage may require the complete traversing of the region in the case of a drone with a camera facing ground or sweeping the floor by a vacuum cleaner.

Consider a robot placed in a large, closed and bounded region is equipped with a sen-

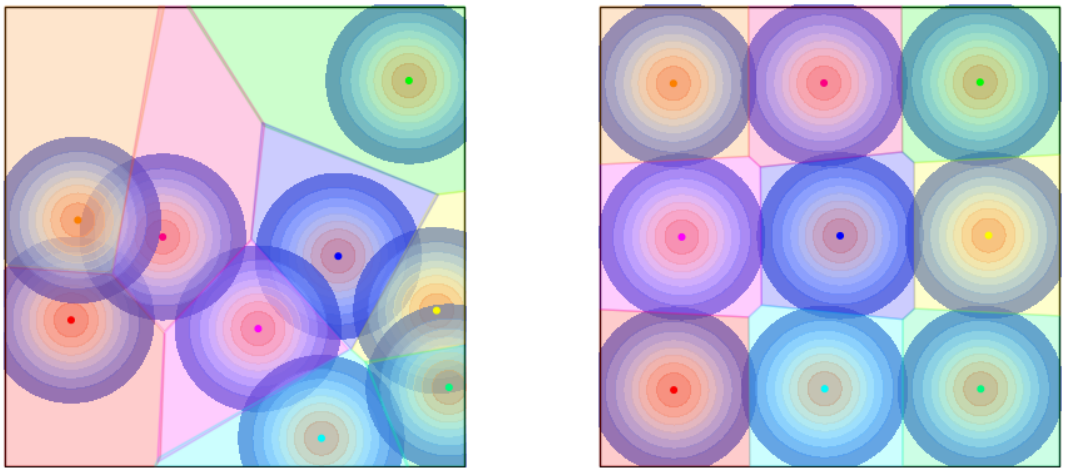
sor. When the robot is stationary, the quality of sensing parts of the region is different. The closer to the robot, the better the sensing will be. In order to achieve proper sensing, the robot has to *move* towards locations poorly sensed previously; hence, doing some amount of work by going the distance. Therefore, the key ingredient to defining the cost is the distance to the locations to be sensed. The actual cost is the summation of such distances over the entire region. For simplicity, a common way to define this cost is to use some distance metric such as Euclidean, Manhattan etc. The main goal is to minimize this distance for each robot deployed in the coverage region.

For generality, the term to be minimized will be referred as the *workload*; local for each agent and global for the overall objective. Minimization of the workload also aims two of the most intuitive goals for groups of mobile robots executing a task: decreasing time-of-completion and energy consumption.

The same idea can be seen in problems like *Locational Optimization* (Okabe and Suzuki, [2]) and *Facility Location Problem* (Drezner and Hamacher, [3]). In these problems, the intuitive goal is to optimize the coverage quality. For example, suppose a set of base stations is to be dispersed over a region. The coverage term, here, becomes the ability of each base station serving the communication needs of the devices nearby. When a client device moves away from a base station, the coverage quality will drop due to the distance that the signals have to travel. Therefore, the goal is to find a set of locations to place the base stations such that the coverage quality would be maximized by minimizing the travelling distance for all signals. When a device moves in the region, it's always served by a base station to a certain level of quality.

2.1.1 Placement vs. Subregion

The intutitions behind placement and subregion concepts can be considered as *duals* to each other. Any placement brings an upper bound of distance cost for coverage, such as in the base station example. When this example is applied over a 2D closed and bounded planar region, one can see that there will be sections in the region, where the cost will not change significantly while moving over them. These are the equidistant lines between base stations, each of which is a *bisector* splitting the region



(a) A random placement

(b) A better placement

Figure 2.1: Two placement schema as a visualization to base station example. Each nested circular groups represent a coverage for the base station placed in the center of them. The coverage degrades as the distance to the base station increases, so the closer to any center, the better. On the left, some sections of the region gets poor service by the stations due to long distances. On the right, however, a better coverage quality over the entire region can be achieved with a more *encompassing* placement. Note that, each section of the region has a lower maximum distance to an assigned station on the right compared to that on the left.

into two subregions. Combining all bisectors and the region boundary will *partition* the region into a set of subregions. Each subregion is assigned to a base station tasked with providing communication services to the devices inside its subregion. On either side of a bisector, the device will always be closer to a base station than the other and the distance to either will be maxed on the line; hence, the upper bound. Now, the goal has become reducing this upper bound with a better placement. Recalling the idea behind the sensing cost, the solution to the CAP problem also relies on an optimal placement of the robots executing the coverage task. This duality is shown in Figure 2.1.

2.1.2 Voronoi Tessellation

Due to being closest in definition with the bisectors partitioning the region, the most popular technique to solve the CAP problem is the *Voronoi Tessellation (VT)*, where each point in the region is assigned to the closest Voronoi generator. In our case, these Voronoi generators are the robot locations in the placement. These point assignments over the entire region results in Voronoi subregions, where Figure 2.1 sets two examples of such partitioning.

Similar to the relationship between placement and subregion concepts, in multi-robot setups, transitioning from one partitioning to another is implemented with a control algorithm that moves the robots to better placements. The advantage of the VT technique is that it can be computed in a distributed manner with ease. While moving, each robot contributes a change into the VT of the region. Upon arrival at a placement such that there are no more significant movements, the solution to the CAP problem is considered to be converged. This logic with the aim of partitioning is seen in *Lloyd's Algorithm*, [1].

2.2 Lloyd's Algorithm in Coverage Control

Suppose N robots are deployed over a coverage region Q with a placement $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}$. Let $\mathbf{V} = \{V_1, V_2, ..., V_N\}$ denote the Voronoi subregions partitioning Q with respect to $\mathbf{P}$; so, $\bigcup_{i=1}^{N} V_i = Q$. A common way to construct the objective for CAP can be described as:

$$H(\mathbf{V}, \mathbf{P}) = \sum_{i=1}^{N} \int_{V_i} f(\mathbf{q}, \mathbf{p}_i)\phi(\mathbf{q})d\mathbf{q} \tag{2.1}$$

where H denotes the cost of coverage. This cost is computed on $\forall \mathbf{q} \in Q$ with f as a function of the distance between $\mathbf{q} \in V_i$ and $\mathbf{p}_i$. The term $\phi(\mathbf{q})$ is commonly considered as a scalar value distribution over Q, indicating some locations may have a different importance for robots to be at. We assume $\phi(\mathbf{q}) = 1$ throughout this thesis.

Lloyd's Algorithm is an iterative optimization scheme solving a least squares optimization problem replacing $f(\mathbf{q}, \mathbf{p}_i)$ with $||\mathbf{q} - \mathbf{p}_i||^2$. The solution to the optimization problem is achieved with gradient descent, where the gradient will indicate the

direction the robots should move opposite to at each iteration. Each robot computes its own gradient in a distributed manner:

$$\begin{aligned}\frac{\partial H(\mathbf{V},\mathbf{P})}{\partial \mathbf{p}_i} &= \int_{V_i} \frac{\partial(||\mathbf{q}-\mathbf{p}_i||^2)}{\partial \mathbf{p}_i}\phi(\mathbf{q})d\mathbf{q} \\ &= \int_{V_i} -2(\mathbf{q}-\mathbf{p}_i)\phi(\mathbf{q})d\mathbf{q}\end{aligned} \tag{2.2}$$

Since integration is linear, splitting this term yields:

$$\frac{\partial H(\mathbf{V},\mathbf{P})}{\partial \mathbf{p}_i} = -2\left(\underbrace{\int_{V_i}\mathbf{q}\phi(\mathbf{q})d\mathbf{q}}_{M_{V_i}\mathbf{C}_{V_i}} - \mathbf{p}_i\underbrace{\int_{V_i}\phi(\mathbf{q})d\mathbf{q}}_{M_{V_i}}\right)$$

where M_{V_i} and $\mathbf{C}_{V_i}$ denote the *mass* (assumed $\phi(\mathbf{q}) = 1$, mass will be equal to the *area* of V_i) and the *centroid* of the subregion V_i, respectively. After simplification, we obtain Equation 2.3.

$$\begin{aligned}\frac{\partial H(\mathbf{V},\mathbf{P})}{\partial \mathbf{p}_i} &= -2(M_{V_i}\mathbf{C}_{V_i} - M_{V_i}\mathbf{p}_i) \\ &= -2M_{V_i}(\mathbf{C}_{V_i} - \mathbf{p}_i)\end{aligned} \tag{2.3}$$

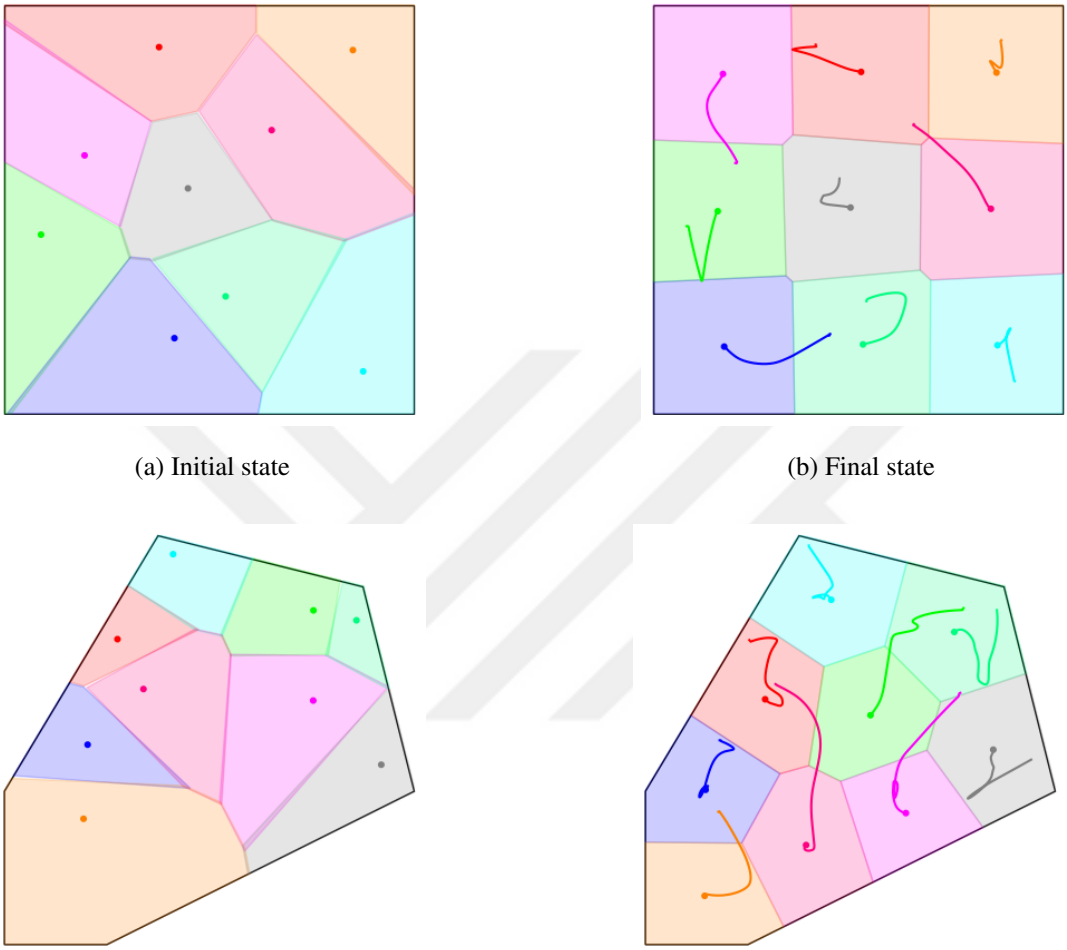
This formulation reduces Lloyd's Algorithm to proportional control law by removing the minus at the beginning for actual gradient descent, where each robot iteratively moves towards a target by a fraction of the error. The fraction will be the proportional gain k_{prop} and the target for robot i will be the centroid of its subregion, $\mathbf{C}_{V_i}$, at any iteration. Then, the control mechanism in Equation 2.4 should drive the robots to the solution placement, $\mathbf{P}^*$, which helps to construct the solution partitioning, $\mathbf{V}^*$.

$$\begin{aligned}\dot{\mathbf{p}}_i &= \mathbf{u}_i \\ \mathbf{u}_i &= k_{prop}(\mathbf{C}_{V_i} - \mathbf{p}_i)\end{aligned} \tag{2.4}$$

Following this strategy, $\mathbf{V}^*$ would be a type of VT where the generators are at the centroids of their subregions, called *Centroidal Voronoi Tessellation (CVT)* (Faber et al., [4]). Figure 2.2 displays CVTs as minimizers for the CAP problem.

2.3 Geometric Properties of Coverage Region

The gradient control in Equation 2.4 works well when the coverage region is *convex*, [1], where no hole exists inside the region and no inner angle is greater than $180°$.



(a) Initial state

(b) Final state

(c) Initial state

(d) Final state

Figure 2.2: Convergence to a CVT with 9 agents each running Lloyd's Algoithm in 2D polygonal regions. The circles and their tracing lines represent the final robot locations and their trajectories, respectively.

However, it is more challenging to partition a non-convex region. Applying Equation 2.4 on a non-convex case results in undesired movements during convergence or final placements at the convergence, as shown in Figure 2.3. To solve this issue, the gradient control should be directed towards reachable targets. Once can accomplish this while also benefiting from the simplicity of Lloyd's Algorithm by changing the distance metric, f, in Equation 2.1.

2.4 Distance Metric in Lloyd's Algorithm

The standard Lloyd's Algorithm uses the squared Euclidean distance as the cost generating a CVT partitioning with:

$$V_i = \{\mathbf{q} \in Q_f | (||\mathbf{q} - \mathbf{p}_i||) \leq (||\mathbf{q} - \mathbf{p}_j||), \forall i \neq j\}$$

$$\Delta_{ij} = \{\mathbf{q} \in Q_f | (||\mathbf{q} - \mathbf{p}_i||) = (||\mathbf{q} - \mathbf{p}_j||), \forall i \neq j\}$$

where Q_f indicates the navigable portion of Q and Δ_{ij} is the bisector shared by V_i and V_j. Q_f will be a strongly connected non-simple polygon with no self-intersections.

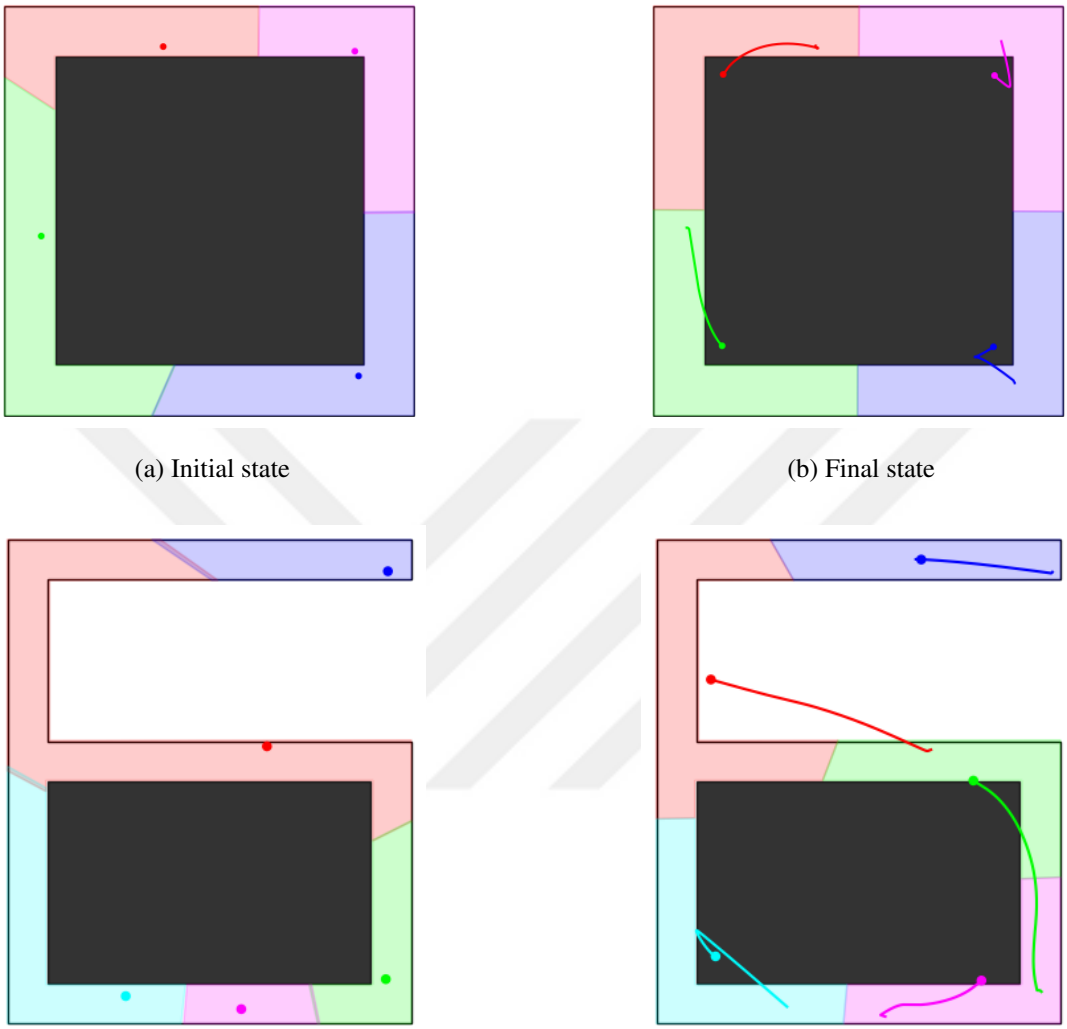
The shortcomings of this setup in non-convex cases are due to the shortcomings of the distance metric. For example, applying Euclidean distance to a non-convex coverage region will cause geometric connectivity issues in subregion assignments. This is shown by Figure 2.4b. To solve this illogical assignment issue we need a distance metric that can satisfy the reachability constraints in non-convex problems. This is where Geodesic distance metric comes in.

Geodesic distance is defined as the length of the shortest path towards the target through the navigable sections of the environment. By definition, it takes reachability into consideration, when computing the bisectors. In this case, the set of points neighbouring robots have equal shortest path lengths to defines the geodesic bisectors. A VT with Geodesic distance metric can be represented as:

$$f_g(\mathbf{q}, \mathbf{p}_i) = ||\mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{p}_i}|| + ||\mathbf{h}^1_{\mathbf{q},\mathbf{p}_i} - \mathbf{h}^2_{\mathbf{q},\mathbf{p}_i}|| + ... + ||\mathbf{h}^l_{\mathbf{q},\mathbf{p}_i} - \mathbf{p}_i||$$

$$V_i = \{\mathbf{q} \in Q_f | f_g(\mathbf{q}, \mathbf{p}_i) \leq f_g(\mathbf{q}, \mathbf{p}_j), \forall i \neq j\}$$

where $h^k_{\mathbf{q},\mathbf{p}_i}$ stands for the k-th waypoint in the shortest path from $\mathbf{q}$ to $\mathbf{p}_i$ and l denotes the number of waypoints in the path excluding the endpoints. The formula for



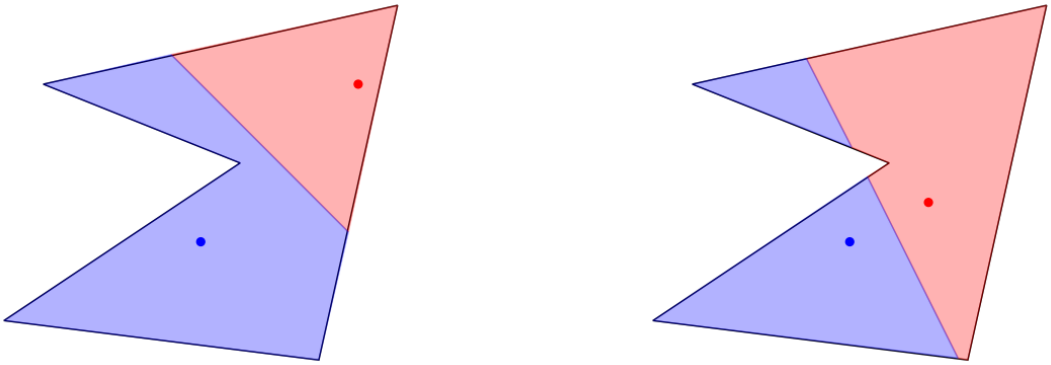
(a) Initial state

(b) Final state

(c) Initial state

(d) Final state

Figure 2.3: The gradient control in the base Lloyd's Algorithm would fail in non-convex environments. The trajectories following the centroids can go through unnavigable sections like holes (obstacles) and boundaries or the partitioning can converge to unnavigable locations.



(a) Connected subregions

(b) Disconnected subregions

Figure 2.4: Two VTs with Euclidean distance metric are applied on a non-convex region. The circles indicate the robot positions. As the robots move closer to each other, there comes an instant, when the bisector starts to intersect with the boundary of the region more than twice. When this happens, disconnected subregions will appear.

geodesic bisectors is similar to the Euclidean using the Geodesic distance. Figure 2.5 shows how the Geodesic distance metric works on a VT.

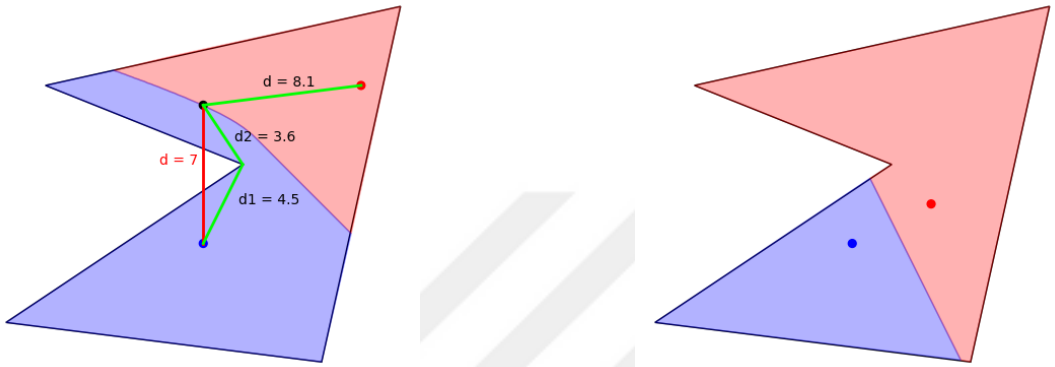
2.5 Geodesic Distance Metric in Lloyd's Algorithm

There are two steps to compute the gradient control: (i) Locating the geodesic centroid and (ii) Calculating the gradient subject to the geodesic path constraint.

2.5.1 Locating the Geodesic Centroid

The geodesic centroid of a region is the location from where the maximum distance to the boundary is minimum. For non-convex polygons without holes, Pollack et al., [5], suggested a method to locate the geodesic centroid with $\mathcal{O}(n \log n)$ time complexity, given the triangulation of the polygon. In addition, they showed that it is *unique*. The complexity is improved by Ahn et al., [6], to $\mathcal{O}(n)$, theoretically. However, neither study considers existence of holes (obstacles in our scenarios).

Under general non-convex circumstances, the geodesic centroid computation is not



(a) A geodesic bisector influenced by the non-convexity

(b) Disconnected piece is attached to the closer robot

Figure 2.5: Applying Geodesic distance metric to Figure 2.4. In (a), the upper section is a more logical partitioning, since the blue robot has to do extra movements to reach the boundary due to the non-convexity. Therefore, the bisector curves into the blue subregion indicating equal arrival effort rather than straight distance. In (b), Geodesic bisector reduces down to a regular Euclidean bisector; however, the disconnected piece in Figure 2.4b is now attached to the red robot, since it's shortest path to that piece is shorter than that of blue robot's.

trivial and practically infeasible. Bae et al., [7] considers the general case in a polygon with holes and suggests a method with $\mathcal{O}(n^{12+\epsilon})$ time complexity using shortest path maps (SPM). With a similar technique, Wang, [8], suggested a method with $\mathcal{O}(n^{11}\log n)$ time complexity.

In our case, the methods in hand are too complicated to implement for exact solutions. Therefore, we consider discretization to shrink the size of the problem so that simpler algorithms can be used.

2.5.2 Discretization of Coverage Area Partitioning

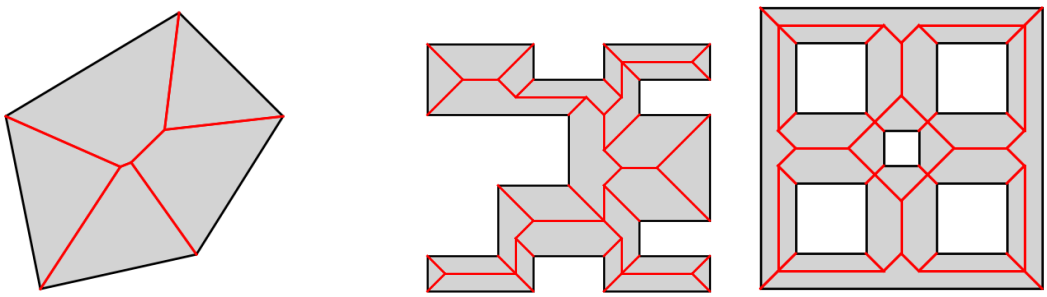
Going forwards we will represent the area as a finite set of navigable locations. We refer to two methods that gives such a set: *Straight Skeletonization* and *Metric Grid* construction.

2.5.2.1 Straight Skeleton of a Polygon

Consider a 2D closed and bounded polygon without any self-intersections. Suppose the edges of the polygon start to move towards inside of the polygon at the same speed and time, keeping parallel to their original orientations. At the beginning, the endpoints of edges neighbouring each other move over their angular bisectors. After some time, edges that are not neighbours before will intersect and form new bisectors partitioning the area in the process. These intersections cause the edges to either grow or shrink in length. Continuing with this process until the edges stabilize in size with no new bisectors to build, we end up with a planar straight line graph, called the *Straight Skeleton* (Aichholzer et al., [9]), is obtained.

The Straight Skeleton of a convex polygon is the VT of the polygon with respect to its edges (also known as Medial Axis Skeleton); however, that is not the case for non-convex polygons. Despite that, the crucial part of the Straight Skeleton of a non-convex polygon is that it provides a *roadmap* for navigation. This roadmap captures the details of reachability between two points in a non-convex polygon. Regardless of whether the polygon is convex or not, in general, the straight skeleton of a simple

polygon is a tree data structure, whereas that of a polygon with holes is a graph. Examples for straight skeletons are shown in Figure 2.6.



(a) Convex - Tree (b) Simple - Tree (c) Non-simple - Graph

Figure 2.6: The red line segments are the edges of the graphs connecting the boundary vertices to the inner vertices. These segments are also the bisectors partitioning the polygon into subregions that display the geometry of the polygon.

Going back to locating the geodesic centroid, with the Straight Skeleton method, we sacrifice the accuracy of exact approaches in favor of an easier implementation. We compute an approximation of the actual geodesic centroid by computing the center of the Straight Skeleton. The center of a planar straight line graph is defined as the node with the minimum eccentricity, which has the maximum distance from that node to any other node in the same graph. We use *Floyd-Warshall*, [10], method to compute all eccentricity values and find the node with the minimum with a $\Theta(n_{ss}^3)$ time complexity, where n_{ss} is the number of nodes in the straight skeleton.

2.5.2.2 Metric Grid of a Polygon

The main weakness of the Straight Skeleton method is the *sparsity* as a planar graph. Sparsity helps representing the problem in a smaller size. However, it drastically sacrifices the accuracy of locating a geodesic centroid in certain cases such as shown in Figure 2.7.

Even though the Straight Skeleton is an approximation, there are cases where we cannot afford such difference in location. In order to make better approximations, a

technique with a higher representational power is required. In other words, a *denser* graph could display more of the geometry of the polygon. To do this, we compute a *Metric Grid* of navigable locations.

The Metric Grid is computed as $G = \{\mathbf{v} \in Q_f | \mathbf{v} = (x_0 + t \cdot r, y_0 + t \cdot r)\}$, where (x_0, y_0) is the lowest position of the bounding box of the environment, $t \in \mathbb{N}$ is a multiplier that generates the grid locations starting from there and $r \in \mathbb{R}$ is the resolution of the grid. Examples to the Metric Grid are shown in Figure 2.8.

2.5.3 Gradient of the Geodesic Lloyd's Algorithm

Having stated how we can locate the centroid, now comes the part to integrate Geodesic distance metric into the gradient control. Regardless of which technique is used for discretization, the idea of geodesic distance remains the same and the following derivation can be applied to both discretization methods. We follow Pavone et al., [11] and Palacios-Gasos et al., [12] for our derivation.

We start with a general workload notation W_i, for simplicity:

$$W_i = \int_{V_i} f_g(\mathbf{q}, \mathbf{p}_i)^2 d\mathbf{q}$$

Then, the partial derivative of Equation 2.1 with respect to $\mathbf{p}_i$ can be written as

$$\frac{\partial H(\mathbf{V}, \mathbf{P})}{\partial \mathbf{p}_i} = \frac{\partial W_i}{\partial \mathbf{p}_i} + \sum_{j \in N_i} \frac{\partial W_j}{\partial \mathbf{p}_i}$$

where W_j terms are computed by other agents locally and collected by communication. By Divergence Theorem, we can rewrite the first term (by local computation)

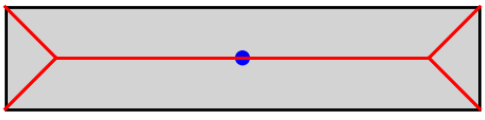
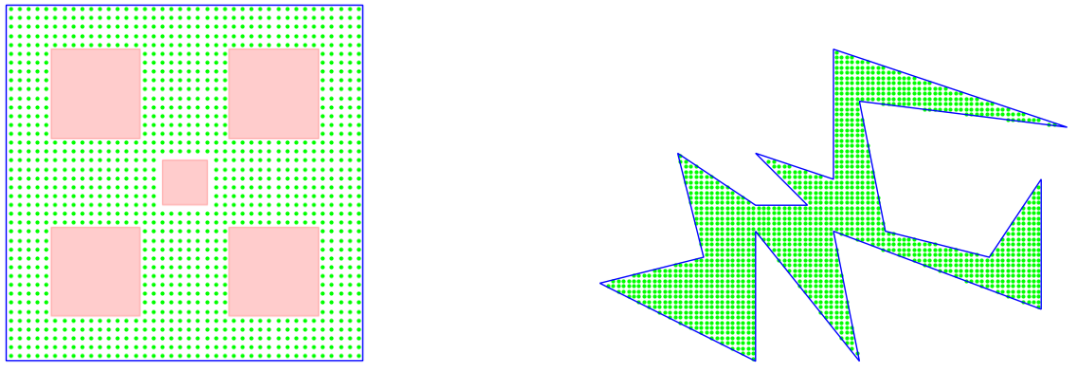


Figure 2.7: Straight Skeleton of a rectangle. If the aspect ratio decreases, the geometry would look more like a square and the center of the Straight Skeleton can approximate the actual centroid (blue dot).



(a) (b)

Figure 2.8: The small green circles represent navigable locations. Note that instead of a sparse navigation graph, we obtain a dense navigation mesh.

as:

$$\frac{\partial W_i}{\partial \mathbf{p}_i} = \sum_{j \in N_i} \int_{\partial V_i} \left(\frac{\partial \mathbf{q}}{\partial \mathbf{p}_i} \mathbf{n}_{\partial V_i}(\mathbf{q}) \right) W_i d\mathbf{q}$$

where ∂V_i is the boundary of the subregion V_i (Figure 2.9a). It has two components: the piece shared with the neighbours, Δ_{ij}, and that shared with the boundary of the navigable region, $\Delta_i^{Q_f}$:

$$= \sum_{j \in N_i} \int_{\Delta_{ij}} \left(\frac{\partial \mathbf{q}}{\partial \mathbf{p}_i} \mathbf{n}_{\Delta_{ij}}(\mathbf{q}) \right) W_i d\mathbf{q} + \underbrace{\int_{\Delta_i^{Q_f}} \left(\frac{\partial \mathbf{q}}{\partial \mathbf{p}_i} \mathbf{n}_{\Delta_i^{Q_f}}(\mathbf{q}) \right) W_i d\mathbf{q}}_{0, \text{ since } \Delta_i^{Q_f} \text{ is static}}$$

where $\mathbf{n}$ in either case stands for the unit outward normal. Notice that the navigable region is static and causes no change in the second term of the split integral; hence, 0. In order to compute the first term, however, we benefit from the definition of the gradient.

The geodesic boundary, Δ_{ij}, as an equality, F, can be written as,

$$F(\mathbf{q}) = f_g(\mathbf{q}, \mathbf{p}_i)^2 - f_g(\mathbf{q}, \mathbf{p}_j)^2 = 0$$

Taking partial derivative of F with respect to $\mathbf{p}_i$ is similar to computing the gradient vector, which is normal to the level curve of F at $\mathbf{p}_i$,

$$\left[2 f_g(\mathbf{q}, \mathbf{p}_i) \frac{(\mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{p}_i}^{l_{\mathbf{q},\mathbf{p}_i}})}{||\mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{p}_i}^{l_{\mathbf{q},\mathbf{p}_i}}||} - 2 f_g(\mathbf{q}, \mathbf{p}_j) \frac{(\mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{p}_j}^{l_{\mathbf{q},\mathbf{p}_j}})}{||\mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{p}_j}^{l_{\mathbf{q},\mathbf{p}_j}}||} \right] \frac{\partial \mathbf{q}}{\partial \mathbf{p}_i} = 2 f_g(\mathbf{q}, \mathbf{p}_i) \frac{(\mathbf{h}_{\mathbf{q},\mathbf{p}_i}^1 - \mathbf{p}_i)}{||\mathbf{h}_{\mathbf{q},\mathbf{p}_i}^1 - \mathbf{p}_i||}$$

where the large term in the square brackets is the gradient and the outward normal, $n'_{ij}(\mathbf{q})$, we need. Then, dividing each side with $||n'_{ij}(\mathbf{q})||$ will give us the unit outward normal,

$$\mathbf{n}_{ij}(\mathbf{q})\frac{\partial \mathbf{q}}{\partial \mathbf{p}_i} = \frac{2f_g(\mathbf{q},\mathbf{p}_i)}{||\mathbf{n}'_{ij}(\mathbf{q})||}\frac{(\mathbf{h}^1_{\mathbf{q},\mathbf{p}_i} - \mathbf{p}_i)}{||\mathbf{h}^1_{\mathbf{q},\mathbf{p}_i} - \mathbf{p}_i||}$$

Putting this term into the integral we obtained earlier results in Equation 2.5.

$$\frac{\partial H}{\partial \mathbf{p}_i} = \sum_{j\in N_i}(W_j - W_i)\int_{\Delta_{ij}} \frac{2f_g(\mathbf{q},\mathbf{p}_i)}{||\mathbf{n}'_{\mathbf{ij}}(\mathbf{q})||}\frac{\mathbf{h}^1_{\mathbf{q},\mathbf{p}_i} - \mathbf{p}_i}{||\mathbf{h}^1_{\mathbf{q},\mathbf{p}_i} - \mathbf{p}_i||}W_i d\mathbf{q} \tag{2.5}$$

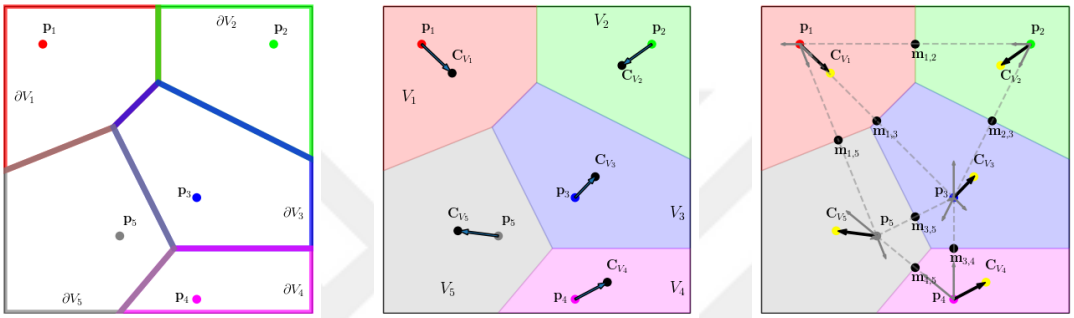
where the term $(W_j - W_i)$ is the difference of workloads as long as agents i and j are neighbouring each other. This is added for *equitable* partitioning. This term prioritizes the directions reducing larger workload differences.

The proportional control now can be represented wih Equation 2.6.

$$\begin{aligned} \mathbf{u}_i^{workload} =& \frac{\partial H}{\partial \mathbf{p}_i} \\ \mathbf{u}_i^{centroid} =& \mathbf{g}_i^* - \mathbf{p}_i \\ \mathbf{u}_i =& k_{goal}(\mathbf{u}_i^{(workload)} + \mathbf{u}_i^{(centroid)}) \end{aligned} \tag{2.6}$$

where $\mathbf{g}_i^*$ is the approximate geodesic centroid of V_i and k_{goal} acts as a proportional gain similar to the standard Lloyd's Algorithm.

The comparison of the Euclidean-driven and Geodesic-driven versions are shown in Figure 2.9.



(a) Boundaries (b) Euclidean-driven (c) Geodesic-driven

Figure 2.9: (a) Overlapping boundaries mean that one robot's movement can affect the other's partitioning. (b) The move-to-centroid behaviour (in black) of the standard Lloyd's Algorithm. (c) Additional forces (in gray), due to boundary sharing in (a), to move the robots to placements, where the partitioning is more equivalent. The $\mathbf{m}_{i,j}$ are middle points for each neighbouring pair.

CHAPTER 3

RELATED STUDIES

In this chapter, we review the previous studies that made use of what we introduced as theoretical preliminaries in the previous chapter. In Section 3.1, we discuss the Voronoi Tessellation (VT) as a solution to partitioning problems. In Section 3.1.1, we review the ideas to enable Lloyd's Algorithm be applicable to non-convex scenarios. Then, Section 3.2 reviews the use of the Geodesic distance metric for such scenarios. Next, in Section 3.3, we discuss the suggestions for equivalent workload partitioning. Finally, Section 3.4 reviews various ideas for discrete solution of area partitioning problem.

3.1 Voronoi Tessellation as Area Partitioning

Cortes et al., [13], applied the standard Lloyd's Algorithm to a 2D convex planar region with a non-uniform value distribution. In their setup, the objective was based on an iterative least squares optimization with the Euclidean distance metric and solved in a distributed manner. They showed that the result is a Centroidal VT (CVT) with the centroid for each subregion is computed by the Shoelace Formula (a special case of Green's Theorem in integration, also known as Gauss Area Formula or Surveyor's Area Formula the latter by Braden, [14]). They showed the adaptivity of Lloyd's Algorithm for area partitioning, even under asynchronous execution.

3.1.1 Locate-and-Move-to-Centroid

In non-convex regions with unreachable centroids, Breitenmoser et al., [15], projected the centroid to the navigable location closest to it. As for arriving at the centroid, Breitenmoser et al., [15], used the Tangent Bug algorithm (Kamon et al., [16]) in non-convex regions. They worked with 2D non-convex regions with uniform value distribution

Palacios-Gasos et al., [12], located the reachable centroid by decomposing the subregion into connected components and selecting the largest component's centroid. They worked on geometries similar to Breitenmoser et al., [15], but included non-uniformity as well.

Instead of directly using motion planning, Klodt et al., [17], used limiting the sensing with visibility constraints. They introduce the Visibility Limited VT to ensure that the targets are directly reachable, even though the visibility constrained subregion can be non-convex. Their experiments are done over 2D non-convex and non-uniform regions.

3.2 Geodesic Distance Metric

Pimenta et al., [18], used the Geodesic distance metric in a heterogeneous multi-agent coverage setup, where sensing capabilities of agents differ. To incorporate this into partitioning, a generalized VT method known as *Power Diagrams (PD)* (Aurenhammer, [19]) is used with squared Geodesic distance. The heterogeneity is applied to the sensing radius of each agent, and then to the weighting factors when computing the boundary equations. Their setup involved convex and non-convex regions with non-uniformity.

3.3 Equitable Partitioning

Pavone et al., [11], showed that when a non-zero variable measure is defined over the coverage region, an equitable Voronoi partitioning generally does not exist, but an

equitable PD does and provided results in convex partitioning problems introducing a regularization term inversely proportional to the variable measure and weights in the boundary equations of PD to shift them correctly.

Palacios-Gasos et al., [12], applied the same idea over non-uniform and non-convex regions for distributed persistent coverage task, where robots continuously move in their subregions to maintain a pre-determined coverage quality value degreading with time. Due to the non-convexity, in addition to Pavone et al., [11], they introduced the Geodesic distance metric to compute PD bisectors.

3.4 Geometric Re-interpretations

3.4.1 Straight Skeleton Method

Aurenhammer, [20], proposed weighted skeletons of simple polygons to test various decomposition and optimality objectives. The weights are associated with the inwards movement speed of boundary segments suring construction, which is equal for all segments in case of a regular straight skeleton. Selbach et al., [21], proposed a method of skeletonization that can adapt to various constraints defined over a simple polygon. Delgado-Frederichs et al., [22], integrated Morse theory to skeletonization of grayscale digital images.

Aichholzer et al., [9], proposed $\mathcal{O}(n^3)$ time and $\mathcal{O}(n)$ space complexity for polygon shrinking. With a priority queue, time complexity was reduced to $\mathcal{O}(n^2 \log n)$, but with $\mathcal{O}(n^2)$ space. Eppstein and Erickson, [23], improved the upper bound to $\mathcal{O}(nr \log n)$, where r is the number of reflex vertices and suggest a time complexity of $\mathcal{O}(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$. On this improvement, Aichholzer and Aurenhammer, [24], suggest that a time complexity of $\mathcal{O}(n)$ is theoretically achievable; however, in practice, the method falls behind the naive approaches due to the use of complicated data structures.

To the best of our knowledge, there are no studies proposing an approach that combines a skeletonization method, the Geodesic distance metric and Lloyd's Algorithm for distributed area partitioning over non-convex and/or non-simple regions, as we do

in this study.

3.4.2 Grid & Graph Methods

Nair and Guruprasad, [25], suggested a grid construction over a non-convex region for a discrete generalized VT. They used the Geodesic Manhattan distance for grid assignment. The Geodesic Manhattan distance is the length of the shortest path where length of each segment between waypoints calculated with Manhattan distance. In the same study, partitioning strategies are also analyzed with respect to the distance functions on complete coverage techniques, as well. These cellular decomposition and graph construction techniques also address the connectivity of the partitioned regions.

Sadeghi et al., [26], suggested a metric graph construction from the sampled locations over a non-uniform and non-convex region. The dispersion measure (i.e. maximum distance between any pair) of sampled locations is constant throughout the region. Their aim was to improve the final result in objective function value obtained by Lloyd's Algorithm with another method. Instead of a move-to-centroid algorithm, they suggested a consensus-like distributed optimization. Each robot sends its neighbours the location in its subregion with the most improvement in objective function and waits for the response of either acceptance of rejection. The retrieval of former means the robot will move and the latter means the robot will look for another local target in its subregion. The communications last until there are no profitable movements and can take place both in synchronous and asynchronous manners.

Broadman et al., [27], applied *Probabilistic Roadmap-Star (PRM*)* (Kavraki et al., [28], then, Kavraki and Kolountzakis, [29]), a sampling based motion planing algorithm, to create a navigation map of the environment. Then, the sampled nodes belonging to this map are partitioned among agents with the Geodesic distance metric under limited and unlimited sensing capabilities. With PRM*, they introduced randomness to the creation of the roadmap.

To the best of our knowledge, studies that use graph construction do use a centroid target in optimization, but do not state the usage of the geodesic centroid. In our case,

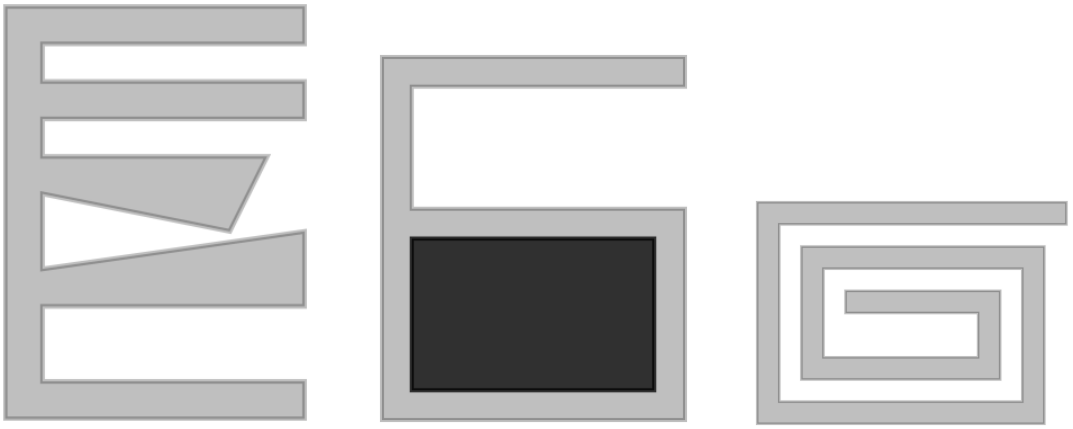
we use a grid construction method for comparsion with the skeleton construction method, both utilizing an approximate geodesic centroid due to the high complexity of computing the exact one.

CHAPTER 4

DISTRIBUTED AREA PARTITIONING FOR COVERAGE

In this chapter, we show our methodology of attacking the CAP problem in general case. In Section 4.1, we breakdown the pipeline of our algorithms into three stages each of which has a common goal using different algorithms. Then, in Section 4.2, we introduce our first method using Straight Skeleton and show its performance under non-convexity. Section 4.3 does the same with our second method using the Geodesic distance metric and Metric Grid. Finally, in Section 4.4, we provide our findings in a more general aspect.

The experiment regions are shown in Figure 4.1. To compare convex performance with standard Lloyd's Algorithm, there will also be an empty square region. These 4 regions will be examined with 4-agent and 8-agent cases with 10 runs each.



(a) Corridors (b) Six (c) Spiral

Figure 4.1: Scenarios for CAP experiments

4.1 Our Methodology

We will describe two distributed algorithms, where each robot follows a *template* shown in Algorithm 1.

Algorithm 1 Template for Agent i

{**– Subregion computation –**}
Broadcast(S_i)
$S_{N_i} \leftarrow$ UpdateNeighbourStates()
$B_{\Delta_{ij}} \leftarrow$ ComputeBisectors(S_{N_i})
$V_i \leftarrow$ ComputeSubregion($B_{\Delta_{ij}}, \partial Q_f$)
{**– Roadmap generation –**}
$G_i \leftarrow$ BuildNavigationGraph(V_i)
$W_i \leftarrow$ ComputeWorkload(G_i)
{**– Gradient control –**}
$\mathbf{g}_i^* \leftarrow$ ComputeCentroid(G_i)
$\mathbf{g}_i' \leftarrow$ ComputeNextTarget($\mathbf{g}_i^*, G_i$)
$\mathbf{u}_i \leftarrow k_{prop} \cdot (\mathbf{g}_i' - \mathbf{p}_i)$
return $\mathbf{u}_i$

We divide this template in 3 stages: (i) Subregion Computation Stage, (ii) Roadmap Generation Stage and (iii) Gradient Control Stage.

4.1.1 Subregion Computation Stage

At this stage, each robot first broadcasts their current state, S_i, which is just the position, $\mathbf{p}_i$, and we assume every robot knows the geometry of the coverage region, particularly the navigable portion, Q_f. Then, each robot computes the pairwise bisectors, $B_{\Delta_{ij}}$, based on updated neighbour states, S_{N_i}. These bisectors are intersected with the region geometry to compute the subregions, V_i.

Two different distance metrics are used for bisector computation: the Euclidean distance (ED) and the Geodesic distance (GD). Standard Lloyd's Algorithm uses ED.

4.1.2 Roadmap Generation Stage

At this stage, each V_i is discretized into local navigation graphs, G_i. We used two methods here: Straight Skeleton (SS) and Metric Grid (MG).

For SS, we used *CGAL* library, [30] with exact computation kernel. The intuition is that a location that has a centroid character can be approximated on a graph that consists of (approximately) equidistant lines.

For MG, we used *Dijkstra*, [31], to compute the bisectors and the navigation graph at the same time; hence, some overlap with previous stage. MG can be considered as a *denser mesh* that can represent the geometry of V_i better than SS, which can provide a better approximation.

The center of mass of a non-convex polygon is likely to be on its exterior. Constraining the selection of the centroid target on a navigation graph helps solving the first half of avoiding the undesired movements, which is to ensure a reachable target.

4.1.3 Gradient Control Stage

At this stage, first, the centroids $\mathbf{g}_i^*$ of G_i are computed with Floyd-Warshall algorithm. Then, the next targets leading to the centroids, $\mathbf{g}_i'$, are computed. This step is the second half of avoiding the undesired movements using motion planning. Next, robots move using the gradient control vector, $\mathbf{u}_i$. We use two types of gradient control: Naive and Equitable. While the Naive version is just the proportional control, the Equitable version prioritizes the largest workload difference among a robot with its neighbours.

4.2 Method 1: ED + SS + Naive Control

This method starts with bisector and subregion computation with ED metric similar to the standard Lloyd's Algorithm, but differs in the second and third stages to solve the navigation issues.

First is the *BuildNavigationGraph* step implemented by Algorithm 2. We build an undirected graph from the raw skeleton, G_i^{SS}, such that G_i represents its symmetric adjacency matrix. Only the bisector edges are taken into consideration, which navigate through inside V_i, and exclude the boundary segments as it hinders the centroid computation. The edge weights are equal to the Euclidean distance between its endpoints.

Algorithm 2 BuildNavigationGraph with SS

$G_i^{SS} \leftarrow$ *CreateStraightSkeleton<CGAL>*(V_i)
$V_{SS}, E_{SS} \leftarrow G_i^{SS}$ {Split into vertex, V_{SS}, and edge set, E_{SS}}
$G_i \leftarrow [0]_{|V_{SS}| \times |V_{SS}|}$ {Adjacency matrix}
for $e \in E_{SS}$ **do**
  $\mathbf{v}_1, \mathbf{v}_2 \leftarrow$ Endpoints(e)
  $G_i(\mathbf{v}_1, \mathbf{v}_2) \leftarrow ||\mathbf{v}_1 - \mathbf{v}_2||$
  $G_i(\mathbf{v}_2, \mathbf{v}_1) \leftarrow ||\mathbf{v}_1 - \mathbf{v}_2||$
end for
return G_i

Secondly, we compute the geodesic centroid with Algorithm 3. Recall that the sparsity advantage of SS is comes with a larger margin in approximating the actual geodesic centroid. In our experiments, this caused an issue with regions whose geometries has similar features to the rectangle in Figure 2.7. To reduce this margin in SS, we added the *WeightedAvg* step which takes the weighted average of two nodes with the least eccentricity using the eccentricity as the weight. This will introduce a new location not in the SS, but approximates better. During our experiments, averaging more than two nodes gave unstable results due to the highly sparse structure of SS.

Algorithm 3 ComputeCentroid for SS

$G_i \leftarrow$ Floyd-Warshall(G_i)
$V \leftarrow$ Vertices(G_i)
$C_i^* \leftarrow$ SortByEccentricity(V, G_i)
$\mathbf{g}_i^* \leftarrow$ WeightedAvg$(C_i^*, 2)$
return $\mathbf{g}_i^*$

Another issue with the sparsity is the uniqueness problem with the geodesic centroid. This can also be seen in Figure 2.7, where there are 2 candidates to be the geodesic centroid, who are the right and left inside intersections. During our experiments, we found that if the SS is a *tree* (for a simple geometry without any holes), then approximating to the unique geodesic centroid is rather successful. When the SS is a graph due to a hole in the region, however, this becomes unlikely.

The third step is the *ComputeNextTarget* step. We have already obtained the navigation graph at the second stage. Therefore, motion planning to a node in the graph is straightforward. The trivial case is SS being a tree, when there is only one path to the target. In the general graph case, methods like Dijkstra and A* can be used.

In our case, however, due to the weighted averaging, we need to plan a path targeting a location not in our navigation graph, $\mathbf{g}_i^* \notin G_i$. We make use of the underlying triangulation used for skeleton construction to compute the *Visibility Polygon* with CGAL, [32], from robot's location $\mathbf{p}_i$. At the *ComputeNextTarget* step, we run Algorithm 4. If $\mathbf{g}_i^*$ is already visible, the robot directly moves there, $\mathbf{g}_i^* = \mathbf{g}_i'$. Otherwise, we tra-

Algorithm 4 ComputeNextTarget for SS

$V_i^{vis} \leftarrow$ *VisibilityPolygon*<*CGAL*>($\mathbf{p}_i$, Q)
if $\mathbf{g}_i^* \in V_i^{vis}$ **then**
  return $\mathbf{g}_i^*$
end if
for $\mathbf{g}_i' \in \mathbf{C}_i^*$ **do**
  if $\mathbf{g}_i' \in V_i^{vis}$ **then**
    return $\mathbf{g}_i'$
  end if
end for

verse the list obtained from the *SortByEccentricity* in Algorithm 3, in ascending order and find the first visible node to set $\mathbf{g}_i'$. In our experiments, if the navigation graph is a tree, then, movement is rather successful. The graph type, on the other hand, requires the shortest path computed from G_i, after *modified* by the Floyd-Warshall method.

4.2.1 Performance

We execute Method 1 for ten runs for each scenario. There are four polygonal regions (i.e. Square, Corridors, Six and Spiral) with two cases (i.e. 4-agent and 8-agent), making 8 scenarios a total of 80 experiments. The stopping condition is that no robot shows a significant movement for the last 20 iterations. For Method 1 this is achieved around 200 - 300 iterations.

Figure 4.2 shows the iterative minimization of standard deviation of the individual workloads at convergence with Method 1; therefore, lower values are expected. Our numeric findings are shown in Table 4.1.

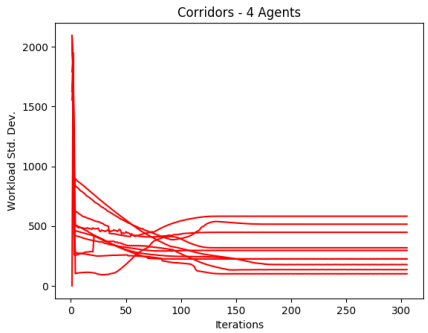
Scenario	Best	Worst	Std. Dev.
Corridors - 4 Agents	100.356	582.329	155.478
Corridors - 8 Agents	127.305	272.152	44.933
Six - 4 Agents	141.783	503.827	124.779
Six - 8 Agents	123.660	290.650	61.441
Spiral - 4 Agents	**8.243**	**79.513**	**21.196**
Spiral - 8 Agents	**22.775**	**60.381**	**12.120**

Table 4.1: Individual workload standard deviation scores of Method 1 in 6 different scenarios. The values for Spiral region are obtained at a coverage percentage less than 100%, as Table 4.2 shows.

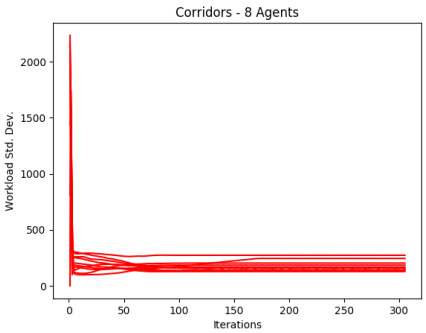
Looking at Figure 4.2, increasing the number of coverage robots helps minimizing the standard deviation; hence, a more equivalent partitioning. Furthermore, we can obtain more homogeneous subregions. This is also supported by the idea that increasing the number of cluster centers will help the clustering procedure to give clusters that are more homogeneous in size.

With Method 1, we can avoid undesirable navigation thanks to the SS in stage two constriaining robots' movements. Figure 4.3 displays the traces of the navigation during experiments for Corridors region compared to the standard Lloyd's Algorithm.

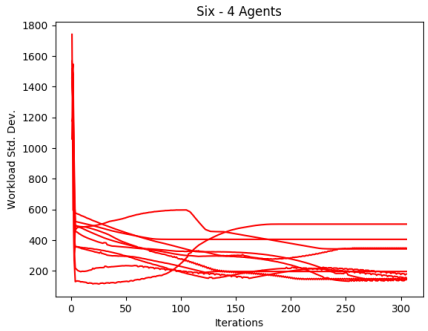
On the other hand, even though Method 1 may be a decent setup for partitioning in



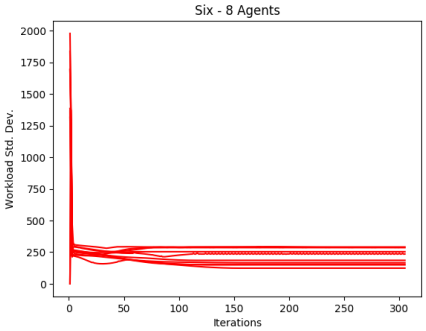
(a) 4 agents



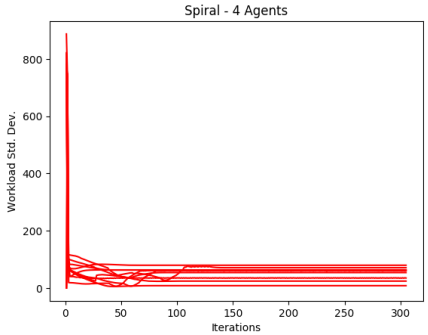
(b) 8 agents



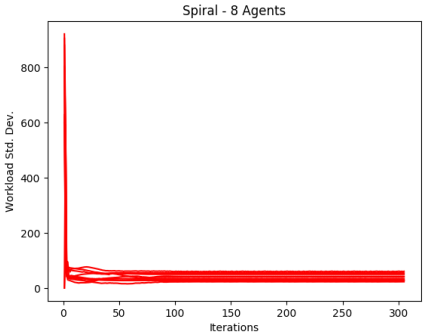
(c) 4 agents



(d) 8 agents

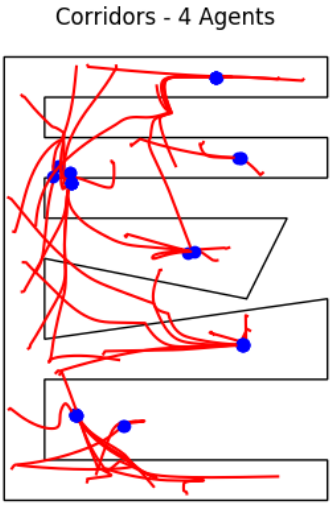


(e) 4 agents

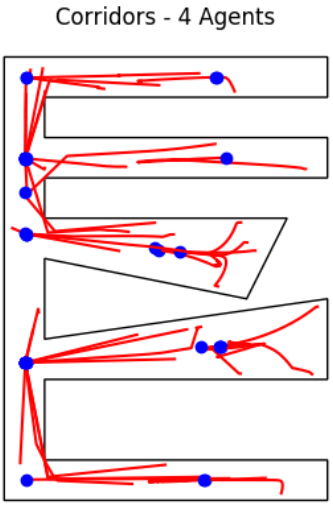


(f) 8 agents

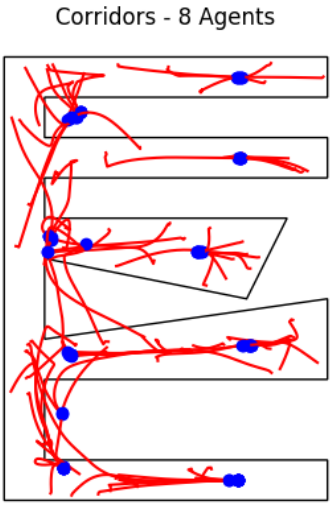
Figure 4.2: Method 1 performance in minimization of individual workload standard deviation over iterations



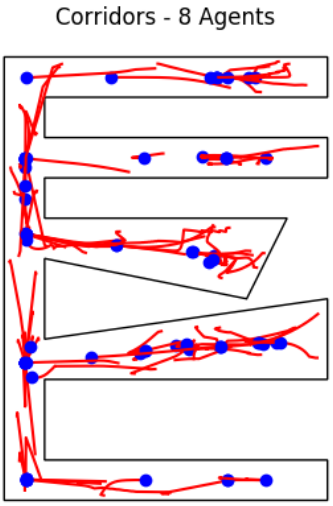
(a) Lloyd - 4 agents



(b) Ours - 4 agents



(c) Lloyd - 8 agents



(d) Ours - 8 agents

Figure 4.3: Navigation traces in Corridors region: Standard Lloyd's Algorithm vs. Method 1. Blue dots indicate the convergence locations over all 10 runs.



(a) Decent

(b) Disconnected

Figure 4.4: Method 1 is not safe from the bias due to the inital placement. Combined with ED in stage one, disconnectivities may happen at any iteration. We do not assign the disconnected subregions for a more logical and practical partitioning.

general, it does not quite achieve the equitable partitioning due to non-zero standard deviation. Furthermore, due to ED in stage one, there can be disconnectivities at final partitionings such as in Figure 4.4. Table 4.2 displays the coverage percentage scores due to this problem.

The solution to the disconnectivity issue is the Geodesic Voronoi Partitioning using the Geodesic distance metric, which we use in our Method 2.

4.3 Method 2: GD + MG + Equitable Control

Computing the first stage with GD is quite complicated due to the involvement of parabolic/hyperbolic curves such as Figure 2.5a. Even though incorporating SS with GD seems to reduce the complexity with the help of sparsity, implementation is not trivial and the results are inaccurate both in approximating the geodesic centroid and the partitioning. Also, more intricate heuristics are required alongside weighted averaging. For ease of implementation, in Method 2 we use MG, instead of SS, sacrificing speed for accuracy.

In this method, we attack the issues of both standard Lloyd's Algorithm and our

Scenario	Best	Worst	Std. Dev.
Corridors - 4 Agents	$\mathbf{1.019}$	0.644	0.127
Corridors - 8 Agents	0.919	0.744	0.065
Six - 4 Agents	0.989	0.815	0.067
Six - 8 Agents	0.993	0.827	0.052
Spiral - 4 Agents	0.612	0.394	0.079
Spiral - 8 Agents	0.771	0.538	0.076

Table 4.2: Coverage percentage scores of Method 1 in 6 scenarios ($\%/100$). Overlaps are allowed to some extent to ease the precision arithmetic within CGAL; hence, the $\mathbf{1.019}$ value.

Method 1. Firstly, we introduce GD in stage one to build the Geodesic Voronoi Partitioning (GVP). To do so, Algorithm 5 is executed in a distributed manner by all robots.

In Algorithm 5, not only we compute the geodesic bisectors $B_{\Delta_{ij}}$, but also construct the navigation graph, G_i^{MG}, as a symmetric adjacency matrix (for convenience, let's call it G_i). Therefore, this algorithm merges the first two stages, GD and MG.

Each robot iteratively expands its wave until either the wave reaches the boundary of the region or intersects with neighbours' waves. In the latter case, we compute the equidistant locations and save the shortest path distance to these locations locally. The expansion is implemented by Algorithm 6 using 4-connectivity. This step is used for synchronization of wave expansions by each robot. Figure 4.5 displays an example for GVP with 4-connectivity.

While the first two stages overlap in one algorithm, the third stage starts similar to that in Method 1. The *ComputeCentroid* step uses Floyd-Warshall algorithm to determine $\mathbf{g}_i^*$ of G_i and keep the shortest path information in a seperate matrix. However, we directly take the node with the minimum eccentricity value and do not apply the *WeightedAvg*, since MG is dense and we do not expect to lose too much accuracy due to high sparsity.

In *ComputeNextTarget*, we use the shortest path matrix from Floyd-Warshall to find

Algorithm 5 WavefrontDijkstra for Agent i

$G_i^{MG} \leftarrow \emptyset$ {MG adjacency matrix to be filled with grid node data}
$B_{\Delta_{ij}} \leftarrow \emptyset$ {Geodesic boundaries shared with neighbours}
while $\exists j \in N_i | F_j \neq \emptyset$ **do**
  $F_i \leftarrow$ Expand(i) {Neighbour expands from old frontier}
  $T \leftarrow \emptyset$ {List of expansions to be deleted}
  for Neighbour $j \in N_i$ **do**
    $F_j \leftarrow$ Expand(j) {Self expands from old frontier}
    for $\mathbf{e} \in F_i^j$ **do**
      if $\mathbf{e}$ is occupied **then**
        $T \leftarrow T \cup \mathbf{e}$
      else
        $G_i^{MG} \leftarrow G_i^{MG} \cup e$
      end if
    end for
    $E_{ij} \leftarrow F_i \cap F_j$ {Intersect neighbouring expansions}
    if $E_{ij} \neq \emptyset$ **then**
      $T \leftarrow T \cup E_{ij}$
    end if
  end for
  $F_i \leftarrow F_i - T$ {Delete occupied}
end while
return G_i^{MG}, $B_{\Delta_{ij}}$

$\mathbf{g}_i'$. The control input for this target is denoted by $\mathbf{u}_i^{centroid}$ in Equation 2.6. In order to approach equitable partitioning, we introduce the workload sensitive force implemented by Algorithm 7, denoted by $\mathbf{u}_i^{workload}$ in Equation 2.6.

4.3.1 Performance

The experiment conditions are the same as Method 1. However, the stopping condition is reduced to the last 5 iterations due to longer durations required for convergence. For Method 2 this is achieved at around 30 - 35 iterations for every region, but the

Algorithm 6 Expand for WavefrontDijkstra for Agent i

$D_i \leftarrow \emptyset$ {Visited set}
$F_i' \leftarrow \emptyset$ {New frontier wave}
while $F_i \neq \emptyset$ **do**
  $\mathbf{v}_i \leftarrow F_i$.pop()
  for $a \in$ [UP, DOWN, LEFT, RIGHT] **do**
    $\mathbf{u}_i \leftarrow \mathbf{v}_i + a$
    if $\mathbf{u}_i$ is not navigable **then**
      Skip this step.
    end if
    $D_i \leftarrow D_i \cup \mathbf{u}_i$
    $G_i^{MG} \leftarrow G_i^{MG} \cup (\mathbf{v}_i, \mathbf{u}_i)$
    $F_i' \leftarrow F_i' \cup \mathbf{u}_i$
  end for
end while
return F_i'

Spiral. In Spiral, we obtained stable results at around 70 - 75 iterations due to local minima.

Figure 4.6 shows the iterative minimization of standard deviation of the workload with Method 2. Our numeric findings are shown in Table 4.3.

Figure 4.6 shows that Method 2 is similar to Method 1 in behaviour. The more agents allocated to coverage task the better the minimization gets. However, particularly in Spiral region, Method 1 could not consistently partition the entire field. On the other hand, Method 2 implements a proper GVP and accomplishes total partitioning. The coverage percentage scores for Method 2 is shown in Table 4.4.

On the other hand, Figure 4.6 also shows that the minimization process encountered several plateaus using our Method 2. We think this is because of the poor tuning of the gradient control inputs, $\mathbf{u}_i^{workload}$ and $\mathbf{u}_i^{centroid}$. We examined that $\mathbf{u}_i^{centroid}$ dominates until the robot arrives or gets close to its current geodesic centroid, after that $\mathbf{u}_i^{workload}$ gets significant enough to cause oscillations in movement (and in minimizations as well).

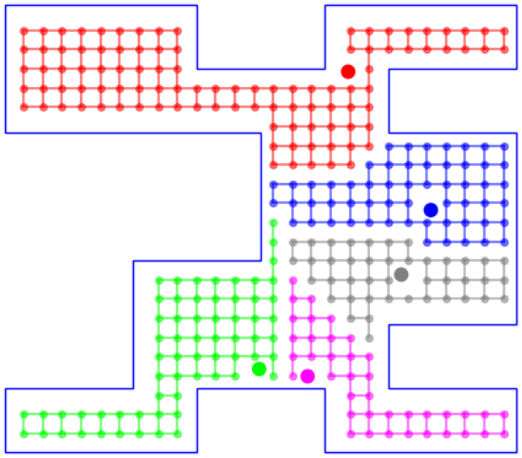


Figure 4.5: A possible GVP with 5 agents using 4-connectivity.

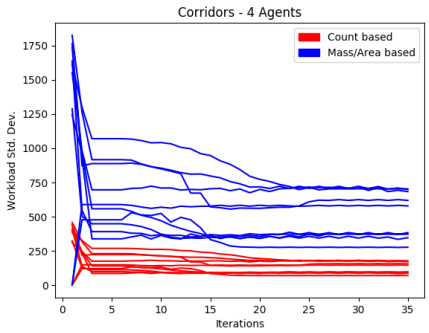
Algorithm 7 WorkloadSensitiveForce for Agent i

$\mathbf{u}_i^{workload} \leftarrow (0,0)$ {Zero vector}
for $\Delta_{ij} \in B_{\Delta_i}$ **do**
  for $\mathbf{q} \in \Delta_{ij}$ **do**
    $\mathbf{u}_i^{workload} \leftarrow \mathbf{u}_i^{workload} + \frac{2f_g(\mathbf{q},\mathbf{p}_i)}{||\mathbf{n}_{ij}(\mathbf{q})||} \frac{(\mathbf{h}^1_{\mathbf{q},\mathbf{p}_i} - \mathbf{p}_i)}{||\mathbf{h}^1_{\mathbf{q},\mathbf{p}_i} - \mathbf{p}_i||}$
  end for
end for
return $\mathbf{u}_i^{workload}$

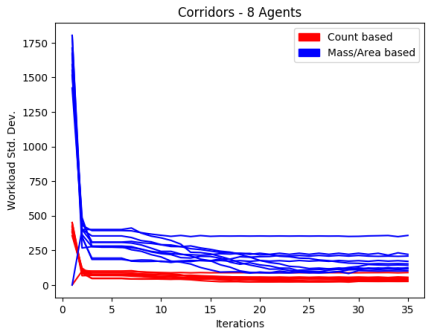
4.4 General Comparison

Figures 4.7a, 4.7b, 4.7c and 4.7d has the following notation: Left and right boxes for each group are for 4-agent and 8-agent cases, respectively. **EL**, **SL** and **GL** indicates Standard Lloyd's Algorithm, Method 1 and Method 2, respectively. *C* and *M* indicates *count* and *mass* (or *area*). The area based results are for comparison, while the count based ones are shown because Method 2 runs over count based computation. Apart from Figure 4.7a, EL does not represent valuable information as it does not work in non-convex regions like the rest of the scenarios.

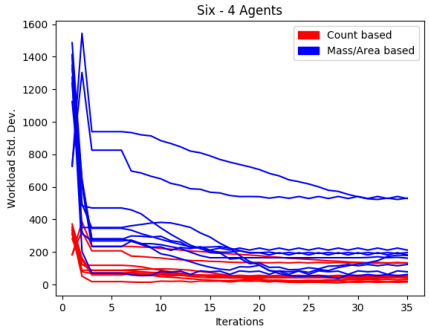
Figure 4.7a shows one of the most ideal cases where the standard Lloyd's Algorithm could be run. It achieves the optimal convergence with 4 agents, but provides no guarantee to do so in the 8-agent case. Method 1 also performs in a similar way, but



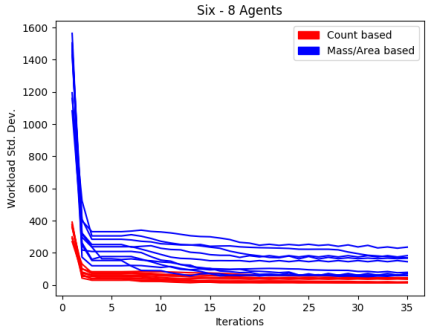
(a) 4 agents



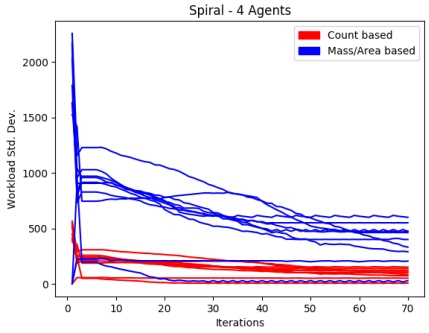
(b) 8 agents



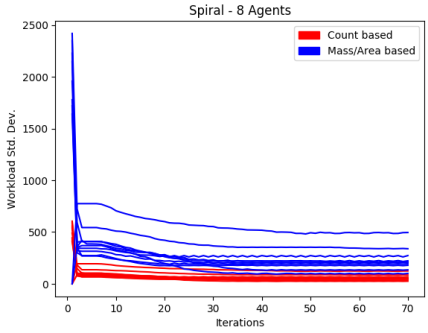
(c) 4 agents



(d) 8 agents

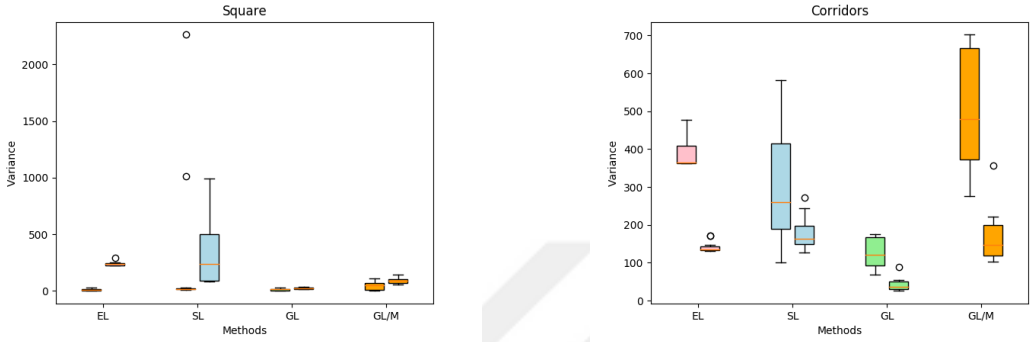


(e) 4 agents



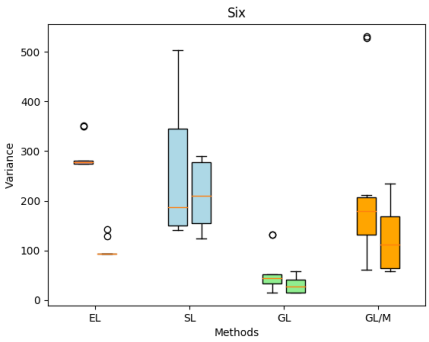
(f) 8 agents

Figure 4.6: Method 2 performance in minimization of workload standard deviation over iterations. The red plots are the standard deviation of the number of navigable locations assigned to each robot, whereas the blue plots are that of the area calculated with $resolution^2 * count$.

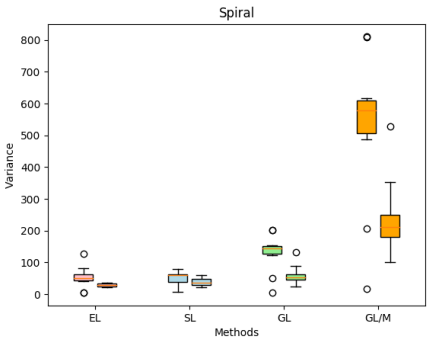


(a) Square

(b) Corridors



(c) Six



(d) Spiral

Figure 4.7: General comparsion of the algorithms with 4-agent and 8-agent cases over the regions

Scenario	Best	Worst	Std. Dev.	Best	Worst	Std. Dev.
Corridors - 4 Agents	69.019	175.540	39.8	276.078	702.159	159.199
Corridors - 8 Agents	25.480	89.372	18.135	101.922	357.490	72.538
Six - 4 Agents	15.188	132.581	39.812	60.754	539.323	159.249
Six - 8 Agents	14.697	58.881	15.235	58.788	235.525	60.939
Spiral - 4 Agents	7.211	149.877	40.831	28.844	599.509	163.322
Spiral - 8 Agents	25.062	123.890	26.803	100.248	495.559	107.213

Table 4.3: Workload standard deviation scores of Method 2 in 6 different scenarios. Left group is based on the number of navigable locations, while right group is area based.

Scenario	Best	Worst	Std. Dev.
Corridors - 4 Agents	0.851	0.837	0.004
Corridors - 8 Agents	0.861	0.843	0.006
Six - 4 Agents	0.986	0.961	0.006
Six - 8 Agents	0.994	0.967	0.008
Spiral - 4 Agents	0.803	0.797	0.002
Spiral - 8 Agents	0.814	0.802	0.004

Table 4.4: Coverage percentage scores of Method 2 in 6 scenarios ($\%/100$).

the 8-agent case is more uncertain. The highly sparse navigation graph is good for fast computation, but does not provide many options in target selection. Method 2 is slower, but achieves similar results in both cases. Therefore, we suggest that Method 2 is better for equitable partitioning.

In Figure 4.7b, we see a different situation. Method 1 in 4-agent case achieves a better partitioning compared to that of Method 2. In 8-agent case, even though Method 2 can get slightly better results, Method 1 is more consistent. We think this is due to the geometry of Corridors region, where each path connecting two navigable locations in separate corridors has to go through the shared section. This can lead to results in Figure 4.8.

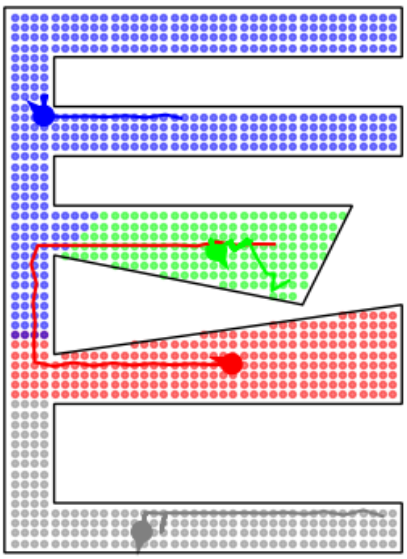


Figure 4.8: Possible partitioning of Corridors region with 4 agents using Method 2. Blue robot occupies most of the shared section. In such cases, the dominance tuning of $\mathbf{u}_i^{workload}$ is poor.

Figure 4.7c shows the opposite of the case in Corridors. In both 4-agent and 8-agent cases, Method 2 can minimize the workload variance better and be at least as consistent as Method 1. The geometry here is not as restrictive as it was in Corridors, where there can be multiple paths going around the large obstacle.

Figure 4.7d shows an edge case with Spiral region. The geometry of Spiral does not allow methods using ED to cover the entire region. In Six and Corridors, Method 1 could achieve a fairly large coverage total, regardless of the equivalence. However, in Spiral, even though Method 1 seems to excel at partitioning, it does so with 61% and 77% coverage in 4-agent and 8-agent cases, respectively. Method 2, on the other hand, provides reliable solutions, albeit inequivalent.

CHAPTER 5

CONCLUSIONS

In this thesis, the distributed area partitioning for multi-robot coverage problem is discussed in terms of what concepts it's built upon. Starting from standard Lloyd's Algorithm, [1], with Centroidal Voronoi Tessellation (CVT), [4], behaviour, step-by-step the major issues are addressed and solved with the help of heuristic methods of re-interpretation of the environment into a topology with Straight Skeleton, [9], and Metric Grid construction methods.

Straight skeleton method addressed the issue of navigating outside of the navigable portion of the environment by constructing a navigation map with the help of Voronoi partitioning of the subregion for each agent. This method overcome the illegal navigation problem in 2 steps. First, the navigation map combined with the Floyd-Warshall method, [10], and an approximate geodesic centroid inside the navigable space by constraining the existence of it to either itself or the navigable neighbourhood surrounding its edges. Then, with the help of motion planning within visibility polygons, its ensured that agents will not deliberately move outside to reach the centroid.

At this stage, weighted averaging with eccentricities as weights is introduced to alleviate the sparsity problem of straight skeletons. Also, since it used standard Voronoi partitioning as its base, straight skeleton also suffered from the issue of unassigned region patches. Finally, Metric Grid method is discussed, where the standard Voronoi is dropped in favor of geodesic Voronoi partitioning computed with Dijkstra, [31], over a grid-like topology where the number of vertices is higher than that of straight skeleton navigation maps'. This introduced additional representational power and capability to capture the non-convexity in a better way.

5.1 Future Work

There are lots of factors in play when distributed area partitioning for coverage in multi-robot setups is considered. Starting with the dimensionality of the task, different concepts and entities are involved going into the higher dimesnions. For instance, in 3D, orientations gain importance for optimal solutions depending on how the robots' sensing capabilities are defined. The complexity increases for a simple technique like Voronoi Tessellation, even more so when Geodesic distance is added. The bisectors can become planes or non-convex surfaces in Geodesic dynamics in higher dimensions.

The control logic of robots is required to conform to the dynamic and kinematic constraints, homogeneous or heterogeneous capabilities etc. In addition, there can be limitations to sensing and communication, uncertainity in movement, sensing or simply failure cases.

As for the objective model, how much importance should be given to metrics like the total completion time, total distance with respect to some metric, the agent network connectivity or the topological connectivity are some of the open-ended parts of the general problem.

In a more abstract way, not knowing the objective model of the global task is another challenge, which is actively studied with Stochastic Optimization, (Kapoutsis et al., [33]). However, before advancing into the hidden representations, the general framework described in Chapter 2, will first be revised covering higher dimensional domains, which will eventually invite the hidden models and inference of them through measurements. The revision is related to the differentiation part, where a robust definition is required for differentiation of abstract topological entitites from graphs, areas, volumes and other simpler structures to more complex ones such as partially defined and smooth hyperbolic curves, hyperplanes, surfaces, or a spatial set of simpler entities such as triangles and so on. If the general framework is well-defined, it can fit into various contexts and be robust and/or invariant to many factors.

REFERENCES

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

[2] A. Okabe and A. Suzuki, "Locational optimization problems solved through voronoi diagrams," *European Journal of Operational Research*, vol. 98, no. 3, pp. 445–456, 1997.

[3] Z. Drezner and H. W. Hamacher, *Facility location: applications and theory*. Springer Science & Business Media, 2004.

[4] V. Faber and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *Siam Review - SIAM REV*, vol. 41, pp. 637–676, 12 1999.

[5] R. Pollack, M. Sharir, and G. Rote, "Computing the geodesic center of a simple polygon," *Discrete and Computational Geometry*, vol. 4, pp. 611–626, 12 1989.

[6] H.-K. Ahn, L. Barba, P. Bose, J.-L. de Carufel, M. Korman, and E. Oh, "A linear-time algorithm for the geodesic center of a simple polygon," 2015.

[7] S. W. Bae, M. Korman, and Y. Okamoto, "Computing the geodesic centers of a polygonal domain," 2015.

[8] H. Wang, "On the geodesic centers of polygonal domains," 2016.

[9] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner, "A novel type of skeleton for polygons," *J. UCS*, vol. 1, pp. 752–761, 01 1995.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "All-pairs shortest paths," in *Introduction to Algorithms, Third Edition*, ch. 25, pp. 693–699, The MIT Press, 3rd ed., 2009.

[11] M. Pavone, E. Frazzoli, and F. Bullo, "Distributed policies for equitable partitioning: Theory and applications," in *2008 47th IEEE Conference on Decision and Control*, pp. 4191–4197, 2008.

[12] J. M. Palacios-Gasós, D. Tardioli, E. Montijano, and C. Sagüés, "Equitable persistent coverage of non-convex environments with graph-based planning," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1674–1694, 2019.

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

[14] B. Braden, "The surveyor's area formula," *The College Mathematics Journal*, vol. 17, no. 4, pp. 326–337, 1986.

[15] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE International Conference on Robotics and Automation*, pp. 4982–4989, 2010.

[16] I. Kamon, E. Rimon, and E. Rivlin, "Tangentbug: A range-sensor-based navigation algorithm," *The International Journal of Robotics Research*, vol. 17, no. 9, pp. 934–953, 1998.

[17] L. Klodt, D. Haumann, and V. Willert, "Revisiting coverage control in nonconvex environments using visibility sets," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 82–89, 2014.

[18] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *2008 47th IEEE Conference on Decision and Control*, pp. 3947–3952, 2008.

[19] F. Aurenhammer, "Power diagrams: Properties, algorithms and applications," *SIAM J. Comput.*, vol. 16, p. 78–96, feb 1987.

[20] F. Aurenhammer, "Weighted skeletons and fixed-share decomposition," *Computational Geometry*, vol. 40, pp. 93–101, 07 2008.

[21] L. Selbach, T. Kowalski, K. Gerwert, M. Buchin, and A. Mosig, "Shape decomposition algorithms for laser capture microdissection," *Algorithms for Molecular Biology*, vol. 16, p. 15, Jul 2021.

[22] O. Delgado-Friedrichs, V. Robins, and A. Sheppard, "Skeletonization and partitioning of digital images using discrete morse theory," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 654–666, 2015.

[23] D. Eppstein and J. Erickson, "Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions," in *In Proc. 14th Annu. ACM Sympos. Comput. Geom*, pp. 58–67, 1998.

[24] O. Aichholzer and F. Aurenhammer, "Straight skeletons for general polygonal figures in the plane," *Lecture Notes in Computer Science*, vol. 1090, 06 1997.

[25] V. G. Nair and K. R. Guruprasad, "Gm-vpc: An algorithm for multi-robot coverage of known spaces using generalized voronoi partition," *Robotica*, vol. 38, no. 5, p. 845–860, 2020.

[26] A. Sadeghi, A. B. Asghar, and S. L. Smith, "Approximation algorithms for distributed multi-robot coverage in non-convex environments," 2020.

[27] B. Boardman, T. Harden, and S. Martínez, "Limited range spatial load balancing in non-convex environments using sampling-based motion planners," *Auton. Robots*, vol. 42, p. 1731–1748, dec 2018.

[28] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[29] L. Kavraki, M. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *Robotics and Automation, IEEE Transactions on*, vol. 14, pp. 166 – 171, 03 1998.

[30] F. Cacciola, "2D straight skeleton and polygon offsetting," in *CGAL User and Reference Manual*, CGAL Editorial Board, 5.3.1 ed., 2021.

[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Single-source shortest paths," in *Introduction to Algorithms, Third Edition*, ch. 24, pp. 658–662, The MIT Press, 3rd ed., 2009.

[32] M. Hemmer, K. Huang, F. Bungiu, and N. Xu, "2D visibility computation," in *CGAL User and Reference Manual*, CGAL Editorial Board, 5.3.1 ed., 2021.

[33] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, “A distributed, plug-n-play algorithm for multi-robot applications with a priori non-computable objective functions,” *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 813–832, 2019.