Distributing work among heterogeneous robots
An approach based on fair division theory

Juan Camilo Gamboa Higuera

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

2012-12-14

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science in Computer Science

DEDICATION

This document is dedicated to my parents, Horacio and Rosa, my brothers, Diego and Andrés, and my closest friend, Diana Grazt.

ACKNOWLEDGEMENTS

I would like to thank all the people who have influenced me during the course of my graduate studies. I want to thank my supervisor Gregory Dudek for his support, encouragement and always insightful comments. I'd like to thank the members of the Mobile Robotics Laboratory: Anqi, Malika, Florian, Yogi, Yiannis, Bikram, Junaed, Patrick, Ian, Pierre Luc, David, Jimmy, Arnold, Sandeep, Dave and Greg; all of them were always open to discuss anything and were very helpful whenever I had a problem. Finally, I'd like to thank my family for their support, the old friends who were always interested in what I was doing: Julián, Daniel, Gonzalo, Diana; and the new friends I made along the way: Marcel, Alex, Liana, Annabel, Ekaterina, James, Philip, Rox and her kids, Bob, among others. If you are one of the people who should have been included here, feel free to be angry at me. I won't forget you for the acknowledgements of my PhD thesis (coming soon!).

ABSTRACT

We study the problem of distributing a single global task between a group of heterogeneous robots. We view this problem as a fair division game. In this setting, every robot defines a *preference* function over parts of the task according to its sensing and motion capabilities. These preferences are described by density functions over the task. We want to find an allocation of the global task that maximizes the probability of task completion. We first formulate the task distribution problem as a fair subdivision problem and provide a centralized algorithm to compute the allocations for each robot. We provide a complexity analysis and computational results of the algorithm. We also provide a decentralized approach, based on the decentralized computations of non-differentiable linear programs using the subgradient method and discuss its convergence properties.

ABRÉGÉ

Nous étudions le problème de distribution d'une simple tâche globale entre un groupe de robots hétérogènes. Nous nous représentons ce problème comme un jeu de division juste. Dans ce contexte, chaque robot définit une fonction de préférence qui règne sur certaines parties de la tâche selon ses capacités de détection et de mouvement. Ces préferences son décrites par des fonctions de densité qui gèrent la tâche. Nous cherchons à trouver une allocation de la tâche globale qui maximise la probabilité de complétion de la tâche. Premièrement, nous formulons le problème de distribution de la tâche comme un problème de subdivision juste auquel on fournit un algorithme centralisé qui calcule les allocations pour chaque robot. Nous fournissons une analyse de complexité et les résultats numériques de l'algorithme. De plus, nous fournissons une approche decentralisée basée sur la décentralisation des calculs de programmes linéaires non-dérivables utilisant la méthode sous-différentielle et discutons ses propriétés de convergence.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Introduction

This thesis considers the problem of coordinating multiple heterogeneous robots to perform a *single global task*, by dividing and allocating the work among them. A task in our context is an activity to be carried out over an operating region. Examples of tasks include cleaning duties inside a building (a coverage task), exploration and mapping of an unknown environment, searching for a target inside a bounded region and aerial surveillance, among others. Although resulting subtasks might have some dependencies between them, we consider the case where progress on the subtasks can be carried out in parallel and independently between the robots.

Coordination of robots through task allocation is motivated by parallelization of workload and robustness to partial failures [1], akin to task scheduling in multiprocessor systems. It has been shown that using multiple robots to perform a single global task results in quantifiable performance gains [2, 3]. These performance gains result from combining the results from single robots into a global result. Aside from parallelization, task subdivision is also motivated by the potential for heterogeneity in the robot team [4], where the assignment of subtasks is dependent on robot abilities.

Existing strategies for task allocation consider the *quality* of the output of the task and an associated *cost* to perform the task for each robot [5]. In this sense, tasks

are considered usually as single atomic unit [6], or as a hierarchy of such units [7, 8]. These single atomic units are usually single points in space or are predefined by a designer or central planner. Although such representation of tasks is useful for the analysis of the Multi Robot Task Allocation (MRTA) problem as a version of the Optimal Assignment problem [9], it might not be easily applicable to tasks that are non-atomic, such as the ones previously mentioned. In this case, subdivision of the task is part of the allocation process.

For the types of tasks targeted by this work, common approaches based on geometric methods, such as polygon area partitioning [10], Voronoi decompositions [11] and approaches based on Ham sandwich cuts [12] and the subdivision is based on characteristics external to the robots; i.e. their position, a distance function, a probability distribution of interest. In this work we are interested in exploiting the heterogeneity of the robot team to drive the task subdivision process.

Heterogeneity of the robot team arises from the workload capacities of each robot and the dependence of the performance of sensing and locomotion on the characteristics of the environment. For example, a legged robot might be slower than a wheeled vehicle on smooth terrain, while on rugged terrain the legged robot has better mobility. Similarly, a robot's size, shape and safety constraints might influence its ability to visit parts of the operating region. Figure 1–1 illustrates an example of a scenario where heterogeneity of a robot team can be exploited. A group of robots is assigned the task of gathering sensory information over a region with multiple kinds of terrain (grass, paved roads) and obstacles (trees, curbs, people). Robots define the regions they "prefer"to work on based on environmental characteristics,

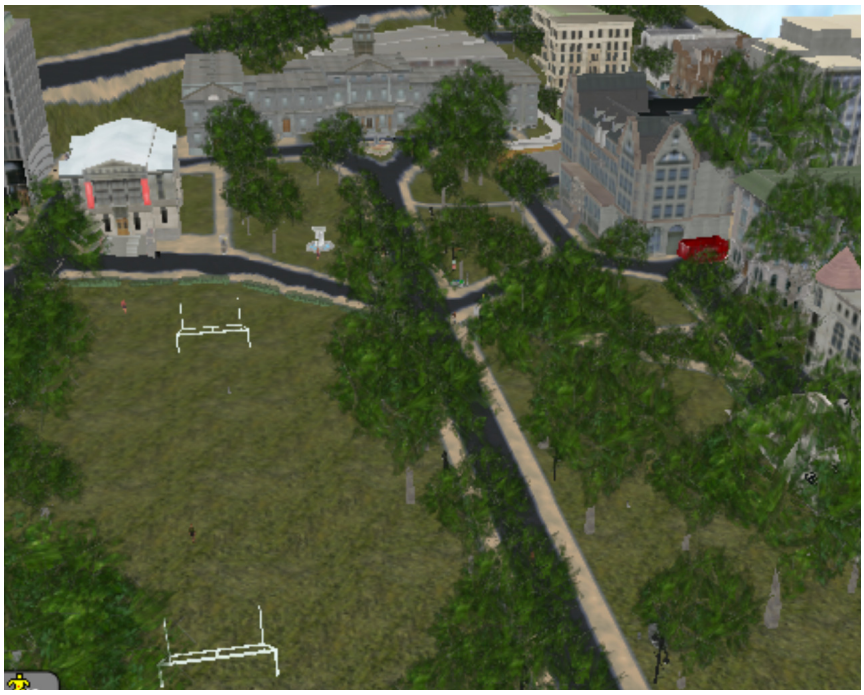


Figure 1–1: Scenario for task subdivision with multiple heterogeneous robots. Robots define the regions they "prefer" based on environmental characteristics. In a search task over this scenario, performance improvements are obtained by assigning the best suited robot for each region. Image generated with the Mammoth engine [13]

such as ruggedness of the terrain and clutter from obstacles. In a search task over this scenario, performance improvements are obtained by assigning the best suited robot for each region; e.g. a ground robot for regions that are covered by trees, an aerial robot for regions that are open, a wheeled robot for regions that are paved, etc. In a sense, this is a way of differentiating between spatially distributed roles for a given task.

For each robot, we encode this dependence with a scoring function that reflects the locations where a particular robot is able to perform some work on the task,

reducing the amount of work left to do. We call the scoring function the robot's *preference*; e.g. a wheeled robot *prefers* smooth terrain. Again, the preference score is based on practical measures of the robot abilities. Accumulating this preference over the allocated subtasks determines a *utility* for each robot. Optimizing these utilities can be interpreted as having robots that "want" to do as much work as possible.

The previous problem can be interpreted as one of fair division: we want to maximize the utility of the task division, while allowing each robot to contribute to the task as much as possible. Fair division theory treats problems of dividing an object between n interested players satisfying some optimality criterion [14, 15]. Applications of fair division theory include land division, multiprocessor scheduling, partition of probability distribution functions and conflict resolution.A common characteristic of fair division problems is that the object to be divided is measurable and every player should receive at least $1/n$ of the object. Two major examples of optimality criteria for fair division are maximizing the minimum utility among the parties (max-min), and maximizing the sum of utilities (max-sum). Maximizing the minimum utility introduces load balancing implicitly, since the solution of the problem will tend to subdivide the task equitably, or at least proportionally [12, 11]. Maximizing the sum of utilities is a simpler problem since it can be solved by assigning parts of the task greedily to the best suited robot. This might result in solutions that assigns subtasks to a subset of robots, which might not be able to complete the task because of physical or energy limitations. Some authors have proposed generalizations of Voronoi diagrams to consider such constraints [16, 17]. In this work we

deal with such cases by stating the fair division problem as a linear program and include the corresponding energy balancing constraints.

Performing task subdivision between mobile robots requires the consideration of communication constraints, leading to situations where the allocation process can only be carried out with distributed information. A well studied approach for distributing work in multi robot systems is that of locational optimization [11, 18, 19, 20, 21, 22]. In such approaches, each robot only needs information from a local neighborhood of robots to obtain an optimal solution. Other approaches for task subdivision have been studied in the multiprocessor scheduling literature [23], based on graph clustering algorithms. For the MRTA problem, an approach based on graph clustering has been proposed, although for atomic tasks [24]. To distribute the task allocation process in our setting, we cluster similar robots together and perform the task allocation recursively inside each cluster. The resulting solution might not be equal to the one obtained by centralized division, but will be approximately the same if the preference functions of robots in different clusters do not overlap. This clustering is inspired by a result in fair division theory, which allows to find optimal divisions by separating players into subgroups with conflicting preferences.

1.2 Contributions of this work

The contribution of this work is the formulation of a task distribution strategy for heterogeneous robots. We present a centralized algorithm that finds globally optimal solutions and propose an extension to a distributed algorithm. What we mean by globally optimal is a solution that, given some performance measures over the allocation for each robot, allocates subtasks that maximize the performance

measures on the task. Example performance measures are the speed of the robots or the expected bits of new information gathered per unit time. Since the space of allocation in our formulation is proven to be convex, we can find a global optimum for a given set of constraints, if enough computation time is available. We also propose how to use this strategy in a multi robot coverage task and show computational results of the impact on the performance of our proposed algorithms when varying parameters such as the number of robots, resolution (cell-size) of the preference functions and similarity between preferences.

1.3 Thesis Outline

This work is organized as follows. In Chapter 2, we provide a description of existing similar and previous work in Multi Robot Task Allocation, Multi Robot Coverage, Territorial Division and Fair division Theory. In Chapter 3, we present the formulation of the task subdivision problem as a linear program. We also describe how to find solutions to this optimization problem and how to extend it to a distributed algorithm. In Chapter 4 we discuss how to apply the fair division theory to a multi robot coverage setting and provide some simulation results. Finally, in Chapter 5 we discuss the results obtained and suggest possible directions for future work.

CHAPTER 2
Background and Related Work

2.1 Introduction

The goal of this thesis is to provide a general method for task subdivision and allocation between a group of heterogeneous robots. To do this, we have based our approach on ideas from multiple fields, which are discussed in the following sections. We start with the existing work in the Multi Robot Task Allocation (MRTA) literature in Section 2.2, since our problem can be viewed as an instance of the MRTA problem. Next, in Section 2.3, we discuss the existing approaches for distributed coverage and exploration, based on region partitioning via Voronoi Diagrams. Related to such approaches, we give an overview of work on division of territories in Section 2.4. Finally, we provide a brief introduction to the fair division, or cake-cutting, problem in Section 2.5.

2.2 Multi Robot Task Allocation

A group of robots is a distributed computing network with mobile processors, dynamic communication links and tasks that require interaction with the physical environment in which the robots operate. Therefore, techniques for distributing workloads between multiple processor systems are related to the techniques necessary to distribute tasks between robots. In fact, this is the view taken when studying the MRTA problem as a version of the Optimal Assignment Problem.

MRTA is the problem of assigning tasks to robots in a way that task completion is guaranteed and some global objective function is optimized. In the simplest version of this problem, robots are capable of performing a single task at any given time and tasks require only one robot for completion. Gerkey and Matarić [5] classify this scenario as ST-SR-IA (Single task robots-Single robot tasks-Instantaneous assignment). More formally, given a set of m tasks $\mathcal{T}$, a set of n robots $\mathcal{R}$, and some performance metric $U : \mathcal{R} \times \mathcal{T} \rightarrow \mathbb{R}^+$, we can define the following linear program,

$$\begin{aligned} \max_{\alpha_{ij}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_{ij} U(i,j) \quad \forall i \in \mathcal{R}, j \in \mathcal{T} \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_{ij} = 1 \qquad \forall j \in \mathcal{T} \\ & \sum_{j=1}^{m} \alpha_{ij} = 1 \qquad \forall i \in \mathcal{R} \end{aligned} \tag{2.1}$$

The set of integers α_{ij} represent an allocation of task i to robot j. Since these parameters are integer, then we are effectively giving a each robot a single task. This kind of optimization problem has been widely studied in the past and can be solved by using the Hungarian algorithm [9] by a centralized planner. Given that communications with a centralized planner might not be possible in various multi robot settings, a distributed approach becomes necessary.

Gerkey and Matarić [25] studied this problem and its dual form which has an economic interpretation, inspiring market-based solutions. Zlot *et al.* [7] proposed a market based algorithm that allows for distribution of hierarchical tasks. In both works, the proposed algorithms are based on *auctioning* mechanisms and broadcast

communications. At any given time, any robot might start an auction of a task it cannot perform. Other robots make bids for the task, based on their local, and private, utility functions. The auctioneer waits for some time to receive offers, after which it assigns the tasks to the highest bidder. New tasks are either introduced by the designer of the system or a centralized planner, generated by a robot from its sensor information or as the result of decomposing a hierarchical task. The benefit of such approaches is that the task subdivision process is robust to partial failures, as long as a failing robot can start an auction with its currently assigned tasks.

Other approaches rely on behavioral architectures. The ALLIANCE architecture , proposed by Parker [26], defines the behaviors and measures of *impatience* and *acquiescence* for each robot. When a robot grows impatient enough, it will overtake a task it perceives is not being accomplished. At the same time, a robot will increase its willingness to give up a task if it perceives it is not progressing on its current task. Both measures are based on the capabilities of the robot. In this architecture tasks are iteratively assigned to the best fit robot that is available.

A similar but simpler approach is the Broadcast of Local Eligibility (BLE) mechanism by Werger and Matarić[27], which is based on a distributed subsumption architecture [28]. For any given task, a robot computes its own local utility in performing a task. This local utility is broadcast to all the other robots within communications range. When a robot receives a local utility estimate from another, the robot compares it with its own estimate. If the robot finds its local utility is higher than the ones received from other robot, it claims the task by inhibiting the

other robots from performing the task. Although simple to implement, these behavior based mechanisms require almost continuous communication between the robots and continuous evaluation of utilities to avoid conflicts in task allocation; conflicting allocations will happen during periods without any communication. As listed by Gerkey and Matarić [5], these approximation algorithms for assignment are (at least) 2-competitive.

Another behavioral approach based on schema theory is the one proposed by Tang and Parker [29]; the ASyMTRe architecture. This approach aims to let a group of robots coalesce into teams with complementary abilities, such that they can perform a task that single robots would not be able to. The connections between complementary abilities that solve a particular task are found by computing a utility which is learned from experience. This utility is a weighted sum between the probability of succeeding in a task and the cost of performing the given task. Complementary abilities are classified into schemas; computational blocks with inputs and outputs that depend on its class. The authors use four classes of schemas: perceptual, motor, environmental sensing and communication schemas. In our case, we consider complementary abilities, but as a way of partitioning the task space into independent subtasks.

Other approaches for task allocation, specifically for the ST-SR-IA class, treat the task allocation problem as a linear optimization problem and propose solutions that do not rely on auction mechanisms, but rather as decentralized computation of partial optimization problems. Atay and Bayazit [6] formulated a combination of *target coverage*, *area coverage*, *exploration* and *communication link maintenance* tasks

as a mixed-integer linear program. The solution of this optimization problem is the set of positions where the robots should move in the next computation step. Every robot solves a version of the optimization problem that only considers information available within its local (1-hop) communication range. In their work, optimal solutions are the positions to which each robot should move so that the area covered by the sensor network, the number of pairs of robots that are able to communicate, the number of targets that are covered by the sensor network, and the number of robots in previously unexplored regions, are all maximized. After computing a locally optimal solution, each robot broadcasts its intended new position (*intentions*), the new positions for the other robots in its local solution (*directives*) and the *assignment* of targets to robots. One issue with this approach is that the time is discretized in a way that requires the robots to have synchronized clocks. If robots do not have synchronized clocks then the convergence properties of the algorithm will change, as certain robots will miss the window for exchanging information over time. The robots use the broadcast information to update their local solution iteratively. The iterative nature of this approach makes it very similar to the BLE mechanism and ALLIANCE architecture. This approach is very useful for mobile sensor networks, but might fail in cases when it is not possible to maintain 1-hop communication links; e.g. in underwater applications.

Liu and Shell [8] developed an algorithm that trades off between a centralized and a distributed approach. Their algorithm is based on the observation that the utility matrix defined by the values $u_{ij} = U(i, j)$ will be sparse in large and heterogeneous robot teams, if the utility of unlikely assignments is set to 0. The authors use

a hypergraph to represent the OAP problem. A hypergraph, defined by $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, consists of a set of vertices $\mathcal{V}$ and a set hyperedges $\mathcal{E}$. Hyperedges can connect more than two vertices in the graph. In their formulation, the vertices represent robots and the hyperedges the relationship between tasks and robots that are able to execute them. To distribute the computation of the OAP problem, the hypergraph is clustered and assignment subproblems are solved within each cluster. Clustering is achieved by applying column and row permutation until an approximately block diagonal matrix is obtained. The solution within a cluster can be fond either by using existing methods or by performing clustering recursively. This approach leads to a solution that approximates the centralized optimum, while distributing the computational load. The recursive partitioning idea has also been exploited in the multiprocessor scheduling literature and is similar to the clustering approach used in this thesis.

Task subdivision without assuming atomic tasks has been studied by Rossi *et al.* [30]. The authors proposed an approach for task subdivision by using pairwise negotiations between robots, based on the theory of Rubinstein negotiations. A general definition of a task is given as follows. A task T is an element of the set $\mathbb{T}$ in which the conjunction $\cup$ and disjunction $\cap$ have been defined. Each task is parametrizable by a vector of k parameters $x \in \mathbb{R}^k$. The authors define a global optimization target as a function of the parameters penalized by the overlap between tasks. Each robot computes a local reward based on the global optimization target and its perceived cost of performing the allocated subtask. To subdivide the work, the robots broadcast the set of parameters of their preferred allocation; termed a

proposal. A robot that receives a proposal can either accept it or make a counteroffer. The initial proposals either assign the whole task to the proposing robot, or include information from previous negotiations. To accept an offer, the reward obtained should be greater than some precomputed target reward and the reward that might be obtained by its own counteroffer. Each time a robot i receives a counteroffer it must decrease the expected reward of its proposal by a *discount factor* $\delta_i \in (0, 1)$. Since there might be multiple ways in which the proposal can be reduced by the desired discount factor, the authors propose using an evolutionary strategy that searches over feasible combinations of parameters and "learns" the other robots discount factors and objective functions from their proposals. In the case of two robots it is shown, from Rubinstein's Bargaining theory, that the process must terminate after a finite number of negotiation rounds, and the reward obtained by each side is determined by the value of the discount factors. If subtasks can be measured as a fraction of the global task, the allocation for the robot that starts the negotiation process is

$$A_1 = \frac{1 - \delta_1}{1 - \delta_1 * \delta_2} \tag{2.2}$$

while the allocation for the counterpart is $A_2 = 1 - A_1$. An extension to three or more negotiating robots is presented in the form of round-robin, one-against-all pairwise negotiations. The motivation for using pairwise negotiations is to let the robots have private information. The only information exchanged between robots is their proposals. While the authors demonstrate the versatility of their task definition two main issues remain: how to pick appropriate discount factors and target rewards. This issue is important because it determines the proportions of the work that each

robot will receive. There is also the question of how this approach compares to centralized allocation or to approaches where the robots share their private objective functions. Although this approach also divides a global task in the process allocation, the shape of allocations is dependent on how the task is parametrized, i.e. a polygonal region is parametrized by its vertices, thus subtasks can only be polygonal regions with the same number of parameters. In our work, shape properties of the allocations are optional and introduced as constraints to the optimization problem.

2.3 Multi Robot Coverage and Exploration

There exists a body of work on approaches to decentralized multi robot control using Voronoi subdivisions of space. The Voronoi diagrams usually define the allocation of work, sometimes implicitly. To drive the robots, centroidal Voronoi updates are used to define provably convergent control rules. The common theme is the deployment of a robot team for distributed sensing, minimizing interference. Cortés *et al.* [31] proposed a distributed coverage approach for convex polygonal environments in which the robots follow a gradient descent law to avoid interference and completely cover a polygon. The final arrangement of the robots is dependent on a probability distribution function. Formally, consider n robots, a convex environment $\Omega \subset \mathbb{R}^N$ with a distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$, a sensor degradation function $f : \mathbb{R}^+ \to \mathbb{R}^+$ and a desired coverage distribution function $\phi : \mathbb{R}^n \to \mathbb{R}^+$. The task is to find an allocation of disjoint subregions A_i for each robot i, whose union is Ω, that maximizes

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{A}) = \sum_{i=1}^{n} \int_{A_i} f(d(p_i, x))\phi(x) \, \mathrm{d}x \tag{2.3}$$

where $\boldsymbol{p} = (p_1, ..., p_n)$ are the robots locations and $\boldsymbol{A} = (A_1, ..., A_n)$ are the allocated partitions. Intuitively, the allocations that maximize the objective function are the ones that position such that all the regions with high value of $\phi(\boldsymbol{x})$ are covered by the closest robot with the best performing sensor. If the robots positions are fixed, the optimal partition is the Voronoi diagram. A Voronoi diagram is defined as the set of allocations $\boldsymbol{A}$ such that

$$A_i = \{x \in \Omega : d(x, p_i) \leq d(x, p_j) \quad \forall j \neq i\} \tag{2.4}$$

The authors demonstrated that to maximize the quantity in Eq. 2.3, the robots only need to update their positions by following the gradient of $\mathcal{H}(\boldsymbol{p}, \boldsymbol{A})$, which can be accomplished by having the robots move in the direction of the centroids of their corresponding partition. The centroid is defined as

$$c_i = \frac{\int_{A_i} x \, \mathrm{d}x}{\int_{A_i} \phi(x) \, \mathrm{d}x} \tag{2.5}$$

By letting each robot move in the direction of $c_i - p_i$, the arrangement of the robots will converge to a centroidal Voronoi partition [32]. Lloyd's gradient descent is a simple algorithm to find centroidal Voronoi partitions. The algorithm consists of:

1. Computing the partitions $\boldsymbol{A}$
2. Computing the centroids for each partition
3. Updating $\boldsymbol{p}$ with the positions of the centroids
4. Step 1).

The authors propose the use of a continuous time version of Lloyd's gradient descent rule as the control law for the robots. To compute its own partition A_i and

centroid, a robot only needs information from its neighbors, allowing for distributed computation.

Multiple extensions to this basic approach have been proposed. For example, Schwager *et al.* proposed an approach that does not assume previous knowledge of the coverage distribution function ϕ. Robots estimate the coverage distribution from local sensing data [33], so that the robots learn which areas need more coverage as they execute the coverage task.

Pimenta *et al.* [21] proposed an extension to non-convex environments by replacing the distance d with a geodesic distance. Bhattacharya and Kumar [18] extended the geodesic distance approach with a method for projecting the centroids of the partitions to admissible locations. To quickly compute an approximate geodesic distance, the authors use Dijkstra's search algorithm. They proposed the use of mapping uncertainty in a occupancy grid as the coverage distribution function. By doing this, the authors were able to drive the robots to explore an unknown environment. This characteristic is similar to the approach by Schwager *et al.*.

Breitenmoser *et al.* used a similar idea, although they do not compute the geodesic distance explicitly. The authors combine gradient descent laws with obstacle avoidance, by means of the TangentBug algorithm [34]. They modify the algorithm to project the centroids of the partitions to admissible locations inside the environment. The authors show that their approach finds local optima for 2.3 constrained to a non-convex boundary.

Although these approaches are related to work subdivision of a single global task, the main difference is that we are interested in matching robot abilities to subtasks, exploiting the heterogeneity between the robots to find allocations.

2.4 Equitable division of territories

Our work is also related to the region partitioning problem [12, 11]. In this problem, an operating region is usually described by a polygon and the goal is to subdivide it into non-overlapping regions. A single facility or vehicle is allocated one such region, on which it provides some service. One of the goals of works on this area is to balance the workload between partitions. Pavone *et al.* [11] developed a polygon partitioning strategy based on a generalization of Voronoi diagram called *power diagrams*. The problem they studied is to divide a territory between multiple facilities, or vehicles, to satisfy a demand distribution. From a set of point $\boldsymbol{p} = \{p_1, ..., p_n\}$ and a set of weights $\boldsymbol{w} = \{w_1, ..., w_n\}$, a power diagram defines a set of partitions $\boldsymbol{A} = \{A_1, ..., A_n\}$ of a polytope $\Omega \subset \mathbb{R}^N$ such that

$$A_i = \{x \in \Omega : d(x, p_i) - w_i \leq d(x, p_j) - w_j \quad \forall j \neq i\} \tag{2.6}$$

The weights have the effect of translating the bisecting line of two neighboring partitions along the line that connects the corresponding centroids. The authors prove that, by fixing the locations of the centroids, the weights of the power diagram can be varied to find a partition that is equitable on an arbitrary, but smooth, measure. Formally, given a measure $\lambda : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^+$ the goal is to find a set of weights such that

$$\int_{A_i} \lambda(x) \, \mathrm{d}x = \frac{M}{n} \tag{2.7}$$

where $M = \int_\Omega \lambda(x)\ \mathrm{d}x$. Such partition is realizable by a power diagram and can be found by performing gradient descent on the set of weights. The authors also provide an extension to allow the centroids to move, resulting in partitions of uniform shape. To account for some heterogeneity between the facilities/vehicles, the authors show that their approach not only finds equitable partitions, but also partitions with some proportional ratios. Our problem can be seen as a version of this one, in which instead of having a single demand distribution to distribute, we have multiple such distributions representing different types of demand and a single facility cannot cover all such types.

Carlsson *et al.* [12] studied the same problem but with a different approach. The idea in this case was to partition two measures over a region of $\mathbb{R}^2$ equitably; e.g. finding partitions of a district with the same amount of people from two political parties, finding partitions of land with equal area and equal populations, or finding partitions of a city each with one postal office and the road length. This idea is based on the *ham-sandwich theorem* which states that an equitable partition of N measures over $\mathbb{R}^N$ always exists. Their approach works by recursively applying *ham-sandwich* cuts to partition the polygon. Figure 2–1 depicts a simple overview of the resulting partition from their algorithm.

Carlsson [35] also studied the problem of dividing a territory among multiple facilities as an infinite dimensional optimization problem. The author studied the properties of solutions which balance the workload between partitions by minimizing the maximum load, minimizing the average load while adding an equal area constraint, and a extension to the previous two solutions that guarantees simply

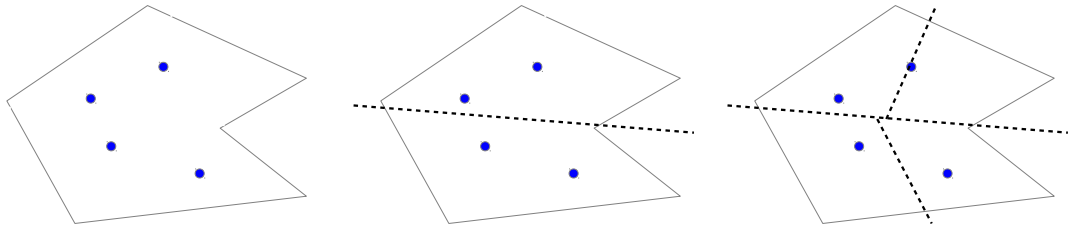


Figure 2–1: Example of equitable division through ham sandwich cuts. (2–1(a)) Region and points to be subdivided. (2–1(b)) First cut results in two regions with equal area and a balanced number of points. (2–1(c)) Recursive cuts in the subregions result in pieces of equal area, each containing a single point.

connected (star shaped) partitions. To find the optimal solutions to his linear programming formulation, the author studied the properties of the dual problem and found that the primal-dual gap is zero and that the dual solutions describe generalizations of Voronoi diagrams. In the case of minimizing the maximum load, the solution is described by a multiplicatively weighted Voronoi Diagram, called Apollonius Diagrams. In the case of equal area partitions which balance the average load, the solutions are described by additively weighted Voronoi Diagrams. The author also proposes a method based on gradient descent for the case when the positions of the facilities are variable and unknown.

Following one of the cases in [35], let us consider the problem of balancing a uniform demand distribution over the unit square. Let Ω be the unit square and $\boldsymbol{A} = \{A_1, \ldots, A_n\}$ the allocated partitions between n facilities. Given the positions the facilities $\boldsymbol{p} = \{p_1, \ldots, p_n\}$ with $p_i \in \Omega, \forall i$, we can define the problem with the

following linear program

$$\begin{aligned} \min_{\mathbb{1}_1(.),..,\mathbb{1}_n(.)} \quad & t \\ \text{subject to} \quad & t \geq \int_\Omega \mathbb{1}_i(x)\|x - p_i\| \mathrm{d}A & \forall i \\ & \sum_{i=1}^{n} \mathbb{1}_i(x) = 1 & \forall x \in \Omega \\ & \mathbb{1}_i(x) \in \{0,1\} & \forall i \in \{1,\ldots,n\} \quad , x \in \Omega \end{aligned} \tag{2.8}$$

where $\mathrm{d}A$ is the infinitesimal area element over Ω and $\mathbb{1}_i(x)$ is an indicator function defined as,

$$\mathbb{1}_i(x) = \begin{cases} 1, & \text{if } x \in A_i \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

By relaxing the integrality constraint on $\mathbb{1}_i(x)$, we obtain the following dual formulation

$$\begin{aligned} \max_{\lambda_1,\ldots,\lambda_n,\Lambda(x)} \quad & \int_\Omega \Lambda(x)\mathrm{d}A \\ \text{subject to} \quad & \Lambda(x) \leq \lambda_i\|x - p_i\| \quad \forall x \in \Omega, i \\ & \sum_{i=1}^{n} \lambda_i(x) \leq 1 \\ & \lambda_i \geq 0 \qquad\qquad \forall i \end{aligned} \tag{2.10}$$

The steps for finding the dual formulation will be described in Section 3.2. An optimal solution for the dual problem in 2.10 will have $\lambda_i > 0, \forall i$ and $\Lambda(x) = \lambda_i\|x - p_i\|$ for some i. This means $\Lambda(x)$ is the lower envelope defined by the function $\Lambda_{max}(x) = \max_i \{\lambda_i\|x - p_i\|\}$ and the allocations will be

$$A_i = \{x \in \Omega : i = \arg\max_j \lambda_j\|x - p_j\|\} \tag{2.11}$$

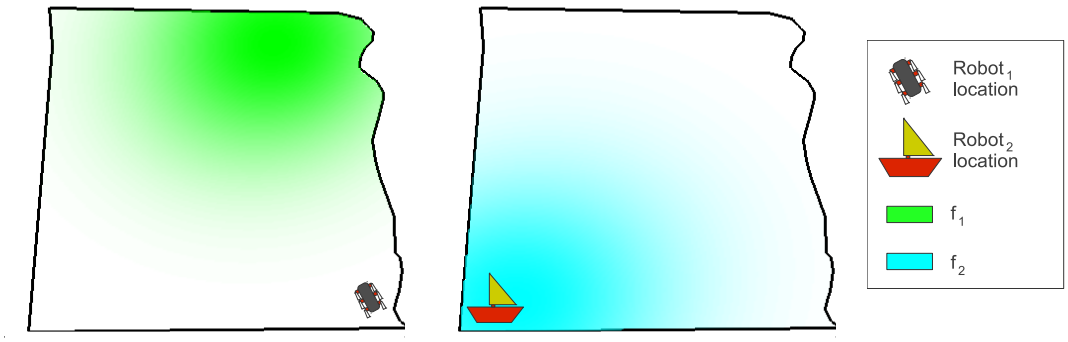
This can be rewritten as

$$A_i = \{x \in \Omega : \lambda_i \|x - p_i\| \leq \lambda_j \|x - p_j\| \quad \forall j \neq i\} \tag{2.12}$$

which is the definition of a multiplicatively weighted Voronoi diagram. Thus with the dual optimization we are finding the optimal ratios of the allocations, defined by the dual variables λ_i, and the function $\Lambda(x)$ will define which points are to be allocated to each partition. We will use a similar approach for solving our problem of task allocation between heterogeneous robots.

2.5 Fair division theory

Fair division is the problem of dividing an object among n interested players, such that each believes it has received a fair share. The players might have different opinions on the value of different parts of the cake; it is this difference in perspective that makes the problem interesting and applicable to our task subdivision problem. A nice historical introduction to fair division theory, in the form of cake cutting problems, can be found in in the book by Robertson and Webb [36]. Figure 2–2 provides an illustration of the problem for dividing a continuous object between 2 players: a region is to be covered with sensors by an autonomous boat and an autonomous underwater vehicle. Within this region, each robot assigns a score to places where it would be better to deploy it, marked by the shading of the region. There are multiple definitions of what fair means which result in somewhat different allocations. A division is said to be *proportional* if it guarantees each player i an allocation of at least a predefined ratio r_i of the total object to be divided, usually $\frac{1}{n}$. A division is *strongly fair* if the allocations are strictly greater than r_i. A division is *envy free* if



(a) (b)

Figure 2–2: Example fair division scenario between two robots.
A six legged robot (2–2(a)) and a boat (2–2(b)). The object to be distributed is a territory depicted by the thick black boundary. For each robot, there is a function f_i that defines how well suited are the robots for performing a given task at any point, which can be viewed as the robot preference. This preference is denoted by the shading of the region: darker regions are the places where the robot is better suited to complete part of the task.

it guarantees that every robot prefers the part it was allocated to that of the other robots; i.e. every robot gets its first choice in the allocation. A division is *Pareto optimal* if the allocation of a single player cannot be increased without reducing the allocation of another player. We are interested in a smaller set of allocations that result from solving linear optimization formulations. The optimal allocations we are looking for are proportional and Pareto optimal. Figure 2–3 illustrates the difference between an optimal and a fair allocation. In the case of the fair allocation, each robot is guaranteed at least a fraction of its maximum allocation; in this case, at least 1/2. In the optimal allocation, every robot is guaranteed allocations which can't be improved locally and each robot will prefer its own allocation to that of any other robot; i.e. the allocations we are interested in are also envy-free. Dubins and

Spanier [14] studied two definitions of optimal partitions, which will be explained here: the max-sum partition and the lex-min partition.

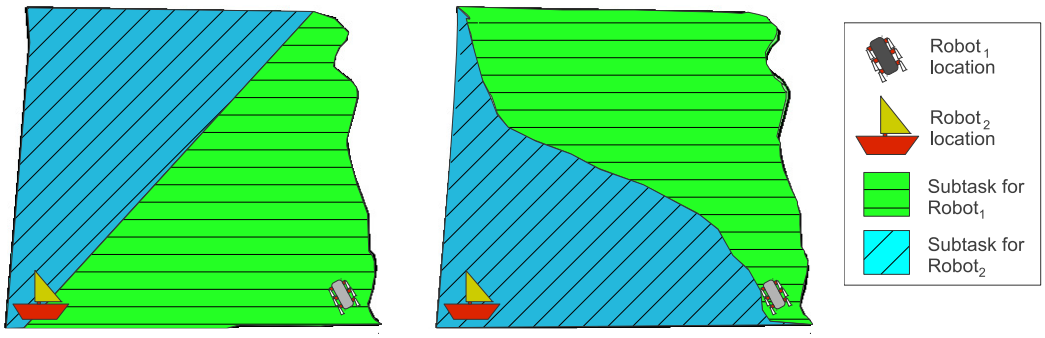
Let the set U represent the object to be divided. Let $u = \{u_1, u_2, \ldots, u_n\}$ be a set of utility functions $u_i : \mathcal{U} \to \mathbb{R}$, where $\mathcal{U}$ is a set of subsets of U which includes U and is closed under complement and countable union operations. Such a set is called a σ-algebra of U. Define a k-partition of U as a set $\boldsymbol{A} = \{A_1, A_2, \ldots, A_k\}$ of elements from $\mathcal{U}$. Each u_i is *countably additive*, i.e. $u_i(A_a \cup a_b) = u_i(A_a) + u_i(A_b)$ for any pair of disjoint sets $A_a, A_b \subseteq U$. In our analysis we will consider utilities such that for every set $A \subset U$ there exists another set $B \subset A$ for which $u_i(A) > u_i(B) > 0$. Such measures are called *non-atomic*. Define the measure v with respect to which each u_i is absolutely continuous. From the Radon-Nikodym theorem, there must exist a function f_i for each u_i such that,

$$u_i(A) = \int_A f_i \, \mathrm{d}v \tag{2.13}$$

where v is a measure with respect to which all u_i are absolutely continuous.

The functions f_i represent a density distribution of the value for player i over the subsets of U. Thus, we can interpret each f_i as the *preference* of player i. The simplest optimization criterion is to maximize the sum of utilities; an optimal partition is an n-partition of U such that $\sum u_i(A_i)$ is maximized. It can be shown that the optimal partition is obtained by a greedy solution. For instance, let $f^* = \max f_i$, an upper envelope of all the measures. Then every n-partition of U is such that

$$\sum_{i=1}^{n} u_i(A_i) = \sum_{i=0}^{n} \int_{A_i} f_i \mathrm{d}v \leq \sum_{i=0}^{n} \int_{A_i} f^* \mathrm{d}v = \int_U f^* \mathrm{d}v \tag{2.14}$$



(a) (b)

Figure 2–3: Example of fair and optimal allocations for the scenario in Figure 2–2. A fair allocation (a) guarantees at least half of the maximum utility, while an optimal partition (b) results in every player obtaining their first choice

Thus the partition that maximizes the sum of utilities is obtained by assigning every subset of U to the player that values it the most. This solution is optimal in that it maximizes the global value, but we might be interested in partitions with other properties. For example, this max-sum solution is Pareto optimal but it is not guaranteed to be proportional or envy free; i.e. a robot that puts a very large value on the full task will be allocated all of it. This leads to the second optimization criterion, which finds partitions by the following scheme:

1. Set $i = 1$, $U' = U$, $k = n$
2. Find the set of k-partitions of U' that maximize the utility of the player that obtains the minimum utility. Call this player p_i and its allocation A_i.
3. Remove p_i from the set of players, make $U' = U' - A_i$ and $k = k - 1$.
4. If $k = 0$, we are done. Otherwise repeat from 2.

Such partitions are called *lexicographic* partitions. A lexicographic partition A^* is optimal if by ordering the utilities in ascending order, for any other partition A'

there is a player p_j for which $u_j(A'_j) < u_j(A^*_j)$. In the case every u_i is non-atomic, Dubins and Spanier proved that an optimal lexicographic partition exists [14]. The optimal partition is also equitable when all the utilities are absolutely continuous with respect to each other. In other words, if all the players assign a non-zero value to every possible subset of U, then the optimal partition will allocate every player the same share. The existence of optimal and equitable partitions is proven by showing that the space of $n \times k$ matrices $M_{ij} = u_i(A_j)$ is convex. As a consequence of the convexity of the space of partitions, proportionally fair partitions must exist (see Corollaries 1.1 and 1.2 in [14]).

Without loss of generality, we will let $u_i(U) = 1$. In general, if every player i is entitled to an allocation of r_i such that $\sum_{i=0}^{n} r_i = 1$ and the measures u_i are non-atomic then there must exist partitions that guarantee each player an allocation of r_i. If at least two players have different preferences then every player will get *at least* r_i. Since the partitions are optimal, all of U for which $f_i > 0$ will be allocated, leading to Pareto optimal allocations. This also applies to the maximum sum allocations. This means that the search space can be reduced to that of Pareto optimal allocations.

Dall'Aglio [15] showed that the space of allocations is indeed convex by using a geometric argument, resulting in a constructive method for finding optimal lexicographic partitions, searching over Pareto optimal allocations. In this method, allocations are parametrized by a set on indicator functions $\mathbb{1}_i$ defined as in (2.9). Thus, the expression in (2.13) can be rewritten as

$$u_i(A_i) = \int_U \mathbb{1}_i(x) f_i(x) v(\mathrm{d}x), \quad \forall x \in U \tag{2.15}$$

Again, we require that all the functions are absolutely continuous with respect to a common measure v. The author provided a method for finding optimal solutions in the dual space and proved that the allocation corresponding to the dual optimum . A summary of the method for a given set of utilities u_i and a set U is as follows.

1. Make $U_1 = U$, $J_1 = 1, \ldots, n$ and $t = 1$
2. Find the simplex vector$\lambda_t \in \Delta_d$ that minimizes $g_t(\lambda) = \int_{U_t} \max_{i \in J_t} \{\lambda_{i,t} f_i(x)\} v(\mathrm{d}x)$
3. If $\lambda_{i,t} > 0$ for all i, go to step 6.
4. Make $t = t+1$, $J_t = \{i \in J_{t-1} : \lambda_{i,t} = 0\}$ and $U_t = U_{t-1} - \{x \in U : \lambda i, t f_i x > 0\}$
5. Repeat from 2.
6. Compute the allocations $\mathbb{1}_i$ as $\{x \in U_t : \arg\max_{i \in J_t} \{\lambda_{i,t} f_i(x)\} \wedge \lambda_{i,t} > 0\}$

The previous method consists of finding subsets J_t of players with conflicting preferences and finding the optimal allocation within each subset. The optimal allocations are the ones that minimize the expression in step 2, as shown in [15], and result in an equitable allocation inside each J_t. Dall'Aglio and Di Luca [37] provided an algorithm that computes the optimal allocation defined by $g_t(\lambda)$ by using the subgradient method [38] and exploiting the fact that optimal allocations are equitable when the utilities are mutually absolutely continuous.

With the tools of fair subdivision it is not to difficult to define the task subdivision and allocation problem as a centralized linear program [15], similar to the approaches cited in Section 2.4. In the next chapter we will provide a formulation of the task subdivision problem that fits nicely with the fair subdivision background. The benefit of using this approach is that we do not need to make any assumptions

about the shape of the object we are trying to divide; i.e. U does not need to be simply connected or polygonal.

CHAPTER 3
Distributing work among heterogeneous robots

3.1 Problem statement

A task $T \in \mathbb{T}$ is assigned to a group of n robots R. The space of tasks $\mathbb{T}$ considered here consists of subsets of the d-dimensional vector space $\mathbb{R}^d$. Each robot is associated with a performance score for each subset of T as the function $f_i : 2^T \rightarrow \mathbb{R}$, where 2^T denotes the power set of T. The goal of our problem is to use the performance functions to encode the heterogeneity of the robots capabilities, and use them to find subdivisions of work which allocate parts of the task to the robots that are best suited for them.

As a motivating example, consider the task of collecting sensor information over some region, represented by $T \subseteq \mathbb{R}^d$ where d is the number of dimensions of the configuration space of the robots. Every robot has both a coverage speed profile $s_i(x)$, which represents its dependence of speed on terrain, and a sensor quality profile $q_i(s_i(x), x)$, which represents the dependence of its sensor performance on location and speed. The speed and quality scores for each location are dependent on each other: if the speed increases then the sensor quality will decrease. Likewise, to increase the quality of the collected data at any one location, the robot must reduce its speed. The robots also have an energy cost dependent on speed and time $c_i(s_i(x), t)$. Not every robot has the same profiles and cost due to differences from design, wear from usage or partial failures. The objective is to maximize both

the speed of coverage by the group of robots and the quality of collected data, while guaranteeing a maximum amount of work done. This means that the ideal distribution of work should avoid interferences/repeated work while assigning the best suited robot to each subregion. Note however that we ignore the case where repeated work might improve quality.

An intuitive approach to do this is to set the speed of coverage to the maximum that guarantees a minimum quality $q_{i,min}(x) = q_i(s_{i,max}(x), x)$ and use $s_{i,max}(x)$ as the quantity to optimize. Robots need power to perform the task thus our method should balance between performing sub optimally and spending energy travelling to remote locations. We also want every robot to do as much work as it can to balance the energy consumption costs. As a first approach, we will consider the problem without direct inclusion of the energy constraints. In Section 3.3.2 we introduce such constraints which do not change too much the method for searching for optimal solutions.

Given the task of collecting sensor data over a region $T \in \mathbb{R}^d$ and a group of n robots, each with its maximum speed profile $s_{i,max}(x)$, we want every part of T to be assigned to the robots in order to minimize the combined time. Formally, let $\boldsymbol{A} = \{A_1, \ldots, A_n\}$ be the allocations of T to each robot $1 \leq i \leq n$. We assume that the time it will take a robot to cover any point $x \in T$ is,

$$\mathrm{d}t(x) = \frac{\mathrm{d}T}{s_{i,max}(x) + \epsilon_s}$$

where $\mathrm{d}T \in T$ is an infinitesimal task element and $\epsilon_s << 1$ is a constant added to avoid division by zero. Thus we want to solve the following optimization problem,

$$\begin{aligned} \min_{A} \quad & \sum_{i=1}^{n} \int_{A_i} \frac{1}{s_{i,max}(x) + \epsilon_s} \mathrm{d}T \\ \text{subject to} \quad & A_i \cap A_j = \emptyset \qquad \forall i, j \in R \\ & \bigcup_{i=1}^{n} A_i = T \end{aligned} \tag{3.1}$$

In this formulation we make the simplifying assumption that the robots are holonomic and are able to compute plans that avoid visiting locations more than once. Therefore the time spent in covering one region is the sum of the times spent at every point in the region. To be able to find solutions to this problem, we require that for every $x \in T$, at least one i exists such that $s_{i,max}(x) > 0$, otherwise x should be removed from T. Letting $f_i(x) = -\frac{1}{s_{i,max}(x)}$, this problem is equivalent to

$$\begin{aligned} \max_{A} \quad & \sum_{i=1}^{n} \int_{A_i} f_i(x) \mathrm{d}T \\ \text{subject to} \quad & A_i \cap A_j = \emptyset \qquad \forall i, j \in R \\ & \bigcup_{i=1}^{n} A_i = T \end{aligned} \tag{3.2}$$

which is in the form of the max-sum fair division problem of section 2.5. This means that the optimal solution is to assign every point $x \in T$ to the fastest robot. In this case, the robot utilities are defined as $u_i(A_i) = -\sum_{i=1}^{n} \int_{A_i} \frac{1}{s_{i,max}(x)+\epsilon_s} \mathrm{d}T$.

A problem with such allocation is that it might end up assigning all of T to a single robot i if it is faster than any other robot; i.e., $s_{i,max}(x) > s_{j,max}(x), \forall j \neq i, x \in T$. To avoid this we can consider energy constraints directly: define the energy

capacity of each robot as $\mathcal{E}_i$ and let $C(A_i)$ be the energy cost for each robot, then add a constraint $C(A_i) \leq \mathcal{E}_i, \forall i \in R$ to the Problem 3.2. Another approach would be to "wrap" the utilities u_i with a concave function, e.g. with a log function, which would have a load balancing effect since the increase in utility would be smaller as the size of the allocation grows. Another simpler approach is to consider the energy constraints indirectly and try to balance the allocations by considering the following max-min optimization problem

$$\begin{aligned} \max_{A} \quad & u_{\min} \\ \text{subject to} \quad & u_{\min} \leq \int_{A_i} f_i(x)\mathrm{d}T \qquad \forall i \in R \\ & A_i \cap A_j = \emptyset \qquad\qquad \forall i, j \in R \\ & \bigcup_{i=1}^{n} A_i = T \end{aligned} \tag{3.3}$$

where $u_{\min}$ is the smallest accumulated score between the robots. The Problem 3.3 is equivalent to the first iteration of finding a lexicographic optimal partition, as explained in Section 2.5. Since our problem is of the form of a fair division problem, the space of allocations is convex and optimal solutions exist. In this formulation we want every robot to do as much work as possible while maximizing the average speed of the slowest robot. With this approach the goal is to be efficient, by assigning regions to the best suited robots, but also fair, by balancing the loads between the robots. These issues will be explored in more depth in the next section.

3.2 Fair Task Subdivision

In this section we present the design of Algorithm 1 (p. 38), which performs task allocation based on the theory of fair division. The problems 3.2 and 3.3 are in the form of fair division problems, thus we know from the work of Dubins and Spanier [14] that an optimal solution must exist. To find optimal solutions we solve the dual optimization problem which is convex, has a finite number of optimization variables and its solutions can be transformed easily into primal solutions, similar to the approaches of Dall'Aglio [15] and Carlsson [35]. To do this, we make use of the indicator function $\mathbb{1}_i$ as described in 2.9. Again, each $\mathbb{1}_i : T \to [0, 1]$ is a function that indicates if an element $x \in T$ belongs to the set A_i. Rewriting Problem 3.3 we get,

$$\begin{aligned} \max_{\mathbb{1}_1,..,\mathbb{1}_n} \quad & u_{\min} \\ \text{subject to} \quad & u_{\min} \leq \int_T \mathbb{1}_i(x) f_i(x) \mathrm{d}T && \forall i \in R \\ & \sum_{i=1}^{n} \mathbb{1}_i(x) = 1 && \forall x \in T \\ & \mathbb{1}_i(x) \in \{0, 1\} && \forall i \in R, x \in T \end{aligned} \tag{3.4}$$

By relaxing the integrality constraint on $\mathbb{1}_i$, i.e. replacing it with $\mathbb{1}_i(x) \geq 0, \forall x \in T$, we can obtain the dual formulation in 3.6. Define the non-negative dual variables $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_n\}$ for each of the constraints of the first group and the function $\Lambda(x)$ for the constraint over the indicator functions $\mathbb{1}_i$. Then modify our optimization problem by introducing the Lagrangian penalization for every constraint,

$$\min_{\boldsymbol{\lambda},\Lambda} \max_{\mathbb{1}_1,..,\mathbb{1}_n} \quad u_{\min} - \sum_{i=1}^{n} \lambda_i \left(u_{\min} - \int_T \mathbb{1}_i(x) f_i(x) \mathrm{d}T \right) - \int_T \Lambda(x) \left(\sum_{i=1}^{n} \mathbb{1}_i(x) - 1 \right) \mathrm{d}T$$

$=$

$$\min_{\boldsymbol{\lambda},\Lambda} \max_{\mathbb{1}_1,..,\mathbb{1}_n} \qquad u_{\min} \left(1 - \sum_{i=1}^{n} \lambda_i \right) + \sum_{i=1}^{n} \int_T \left(\lambda_i f_i(x) - \Lambda(x) \right) \mathbb{1}_i(x) \mathrm{d}T + \int_T \Lambda(x) \mathrm{d}T \tag{3.5}$$

By rearranging terms we obtain the second expression in 3.5. We can obtain the dual formulation 3.6, by noticing that $(f_i(x) - \Lambda(x)) \leq 0$ for the maximization to be feasible.

$$\begin{aligned} \min_{\boldsymbol{\lambda},\Lambda} \quad & \int_T \Lambda(x) \mathrm{d}T \\ \text{subject to} \quad & \Lambda(x) \geq \lambda_i f_i(x) \quad \forall i \in R, x \in T \\ & \sum_{i=1}^{n} \lambda_i = 1 \qquad \forall i \in R, x \in T \\ & \lambda_i \geq 0 \qquad\qquad\qquad \forall i \in R \end{aligned} \tag{3.6}$$

From the first constraint and the objective function it is easy to note that the optimal $\Lambda(x) = \Lambda^*(x)$ should attain the equality $\Lambda^*(x) = \lambda_i^* f_i(x)$ for some i. If any $\lambda_i = 0$ then $\Lambda(x) \geq 0, \forall x \in T$ and the objective function would be minimized with $\Lambda^*(x) = 0, \forall x \in T$, therefore every λ_i should be strictly positive. By complementary slackness [35], the optimal primal solution is such that $u^*_{\min} = \int_T \mathbb{1}_i(x) f_i(x) \mathrm{d}T, \forall i \in R$; i.e. every robot should get the same utility, taking the same time to complete their share

of the task. The previous problem can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & \int_T \max_i \{\lambda_i f_i(x)\} \mathrm{d}T \\ \text{subject to} \quad & \sum_{i=1}^{n} \lambda_i = 1 \qquad\qquad \forall i \in R, x \in T \\ & \lambda_i \geq 0 \qquad\qquad \forall i \in R \end{aligned} \tag{3.7}$$

This dual problem is convex, so we can use an iterative search algorithm to find the optimal values for λ_i. Let $g(\boldsymbol{\lambda}) = \int_T \max_i \{\lambda_i f_i(x)\} \mathrm{d}T$, i.e. the objective function we are trying to minimize. Let $D_{\boldsymbol{\lambda}}$ be its domain. The function g is equivalent to the one in the minimization step of the algorithm in Section 6 of [15]. Since the objective function in 3.7 is not differentiable we may use a subgradient search method. Such a method consists of the following update rule,

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} - \alpha_t \boldsymbol{\gamma}(\boldsymbol{\lambda}^{(t)}) \tag{3.8}$$

where t is the iteration index and α_t is the step size. Let $\boldsymbol{\lambda}^{(t)}$ be an interior point of $D_{\boldsymbol{\lambda}}$. The vector $\boldsymbol{\gamma}(\boldsymbol{\lambda}^{(t)})$ is called a subgradient of g around $\boldsymbol{\lambda}^{(t)}$ that satisfies

$$g(\boldsymbol{\lambda}^{(t+1)}) - g(\boldsymbol{\lambda}^{(t)}) \geq (\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)})^T \boldsymbol{\gamma}(\boldsymbol{\lambda}^{(t)}), \qquad \forall \boldsymbol{\lambda} \in D_{\boldsymbol{\lambda}} \tag{3.9}$$

i.e., the subgradient defines a hyperplane that supports the set of feasible solutions around the point $\boldsymbol{\lambda}$. To find an expression for a subgradient of g, we define an indicator $\mathbb{1}_{i,max}^{(t)}$ as

$$\mathbb{1}_{i,max}^{(t)}(x) = \begin{cases} 1, & \text{if } \lambda_i^{(t)} f_i(x) \geq \lambda_j^{(t)} f_j(x) \qquad \forall j \neq i \\ 0, & \text{otherwise} \end{cases}$$

and rewrite 3.9 as

$$\begin{aligned} g(\boldsymbol{\lambda}^{(t+1)}) - g(\boldsymbol{\lambda}^{(t)}) &= \int_T \max_i \{\lambda_i^{(t+1)} f_i(x)\} - \int_T \max_i \{\lambda_i^{(t)} f_i(x)\} \\ &= \sum_{i=1}^{n} \int_T \mathbb{1}_{i,max}^{(t+1)}(x) \lambda_i^{(t+1)} f_i(x) - \sum_{i=1}^{n} \int_T \mathbb{1}_{i,max}^{(t)}(x) \lambda_i^{(t)} f_i(x) \\ &\geq \sum_{i=1}^{n} \lambda_i^{(t+1)} \int_T \mathbb{1}_{i,max}^{(t)}(x) f_i(x) - \sum_{i=1}^{n} \lambda_i^{(t)} \int_T \mathbb{1}_{i,max}^{(t)}(x) f_i(x) \end{aligned}$$

where we have used the facts that g is convex and $\mathbb{1}_{i,max}^{(t+1)}$ maximizes the value of the integrand of g for a given point $\boldsymbol{\lambda}$. Therefore, we can set the subgradient to be the vector with components defined as

$$\gamma_i^{(t)} = \int_T \mathbb{1}_{i,max}^{(t)}(x) f_i(x) \mathrm{d}T \tag{3.10}$$

which is verified to be a subgradient of g since 3.9 holds. To deal with the constraint $\sum_{i=1}^{n} \lambda_i = 1$, the subgradient method must be modified by including a normalizing constant ν,

$$\sum_{i=1}^{n} \lambda_i^{(t+1)} = \sum_{i=1}^{n} (\lambda_i^{(t)} - \alpha_t \gamma_i^{(t)} + \nu) = 1$$

from which it is easy to see that $\nu = -\alpha_t \frac{\sum_{i=1}^{n} \gamma_i^{(t)}}{n}$. Thus the subgradient update rule for the constrained problem is

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} - \alpha_t \left(\gamma_i^{(t)} - \frac{\sum_{i=1}^{n} \gamma_i^{(t)}}{n} \right) \tag{3.11}$$

To keep every $\lambda_i^{(t+1)}$strictly positive then the step length α_t should satisfy

$$\alpha_t < \frac{\lambda_i^{(t)}}{\left(\gamma_i^{(t)} - \frac{\sum_{i=1}^{n} \gamma_i^{(t)}}{n} \right)}$$

for the cases when $\gamma_i^{(t)} > \left(\sum_{i=1}^{n} \gamma_i^{(t)}\right)/n$. This constraint provides an upper limit for the step length at any time step, which is enforced in step 21 of the Algorithm 1 by multiplying the upper limit by a constant $\eta < 1$. The choice of the step length α_t in 3.11 determines the rate of convergence and the accuracy of the solution. In particular, as shown in [38], if we want the method to converge to the optimal solution then α_t should follow a *diminishing step length rule*. Such rule requires the step size to satisfy

$$\begin{gathered} \alpha_t = \frac{\alpha'_t}{\|\gamma_i^{(t)}\|} \\ \alpha'_t \geq 0, \quad \lim_{t\to\infty} \alpha'_t = 0, \quad \sum_{i=1}^{\infty} \alpha'_t = \infty \end{gathered} \tag{3.12}$$

Examples of sequences that satisfy 3.12 are of the form $\alpha'_t = \alpha'_0/t^p$, where $0 < p \leq 1$. Such sequences are called as *p-series*, which are known to be divergent; i.e. $\lim_{k\to\infty} \sum_{i=1}^{k} \alpha'_t \to \infty$. The choice of p for these sequence will determine the speed of convergence and the error at convergence. For our experiments we set $p = 1$. Subgradient methods are usually used without any formal stopping criteria since these are problem specific. Dall'Aglio and Di Luca [37] provide two stopping criteria for the subgradient method for fair division: until $\max_{j\in R} u_j(A_j) - \min_{j\in R} u_j(A_j) < \epsilon$ or by checking the distance between geometric upper and lower bounds on the optimal set. Both are based on the observation that the optimal solutions for the fair division problem are equitable when the utilities u_i are normalized, mutually absolutely continuous and linearly independent. The utilities are mutually absolutely continuous if for every subset $A \subseteq T$, there exists at least one robot for which the utility $u_i(A)$ is positive. The utility functions in our case are not normalized, so the

stopping criterion is

$$\frac{\max_{j\in R} u_j - \min_{j\in R} u_j}{\max_{j\in R} u_j} < \epsilon$$

where $u_j = \int_T \mathbb{1}_i f_i(x)\mathrm{d}T$.

Algorithm 1 describes the procedure to compute the time optimal solution for the multi robot problem described in Section 3.1. This algorithm assumes that every robot has enough energy complete its part and ensures that the spatial load is balanced between robots. Since the subgradient is not necessarily a descent direction we must keep track of the best solution found so far,

$$\boldsymbol{\lambda}_{best}^{(t+1)} = \arg\min_{\boldsymbol{l}} g(\boldsymbol{l}), \quad \boldsymbol{l} \in \{\boldsymbol{\lambda}_{best}^{(t)}, \boldsymbol{\lambda}^{(t+1)}\} \tag{3.13}$$

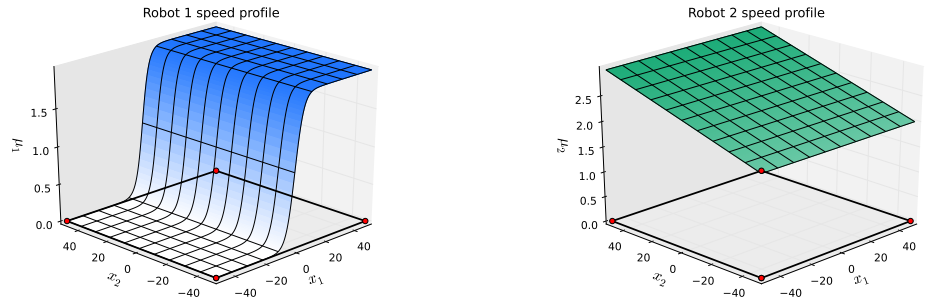
which is done in steps 18 to 20 of Algorithm 1.

3.3 Experimental Results

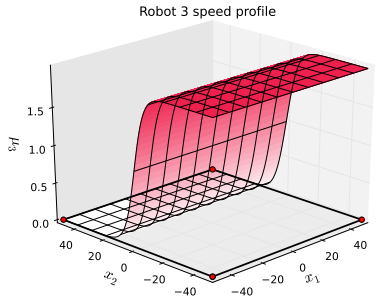
An example of the execution of this algorithm is shown in Figure 3–2, for the task of covering a square region of $100m \times 100m$ using three robots with speed profiles defined as

$$s_1(x) = \frac{2}{1 + e^{-40x_1+20}} \quad m^2/s$$
$$s_2(x) = 2 + \frac{1}{100}x \quad m^2/s$$
$$s_3(x) = \frac{2}{1 + e^{30x_2-15}} \quad m^2/s$$

an integration cell size of $0.0625m$, and a error tolerance ϵ of 0.000001. The final allocated areas are $a_1 = 2966.8m^2$, $a_2 = 4087.2m^2$ and $a_3 = 2946.0m^2$. The final utilities are $u_1 = -1487.2s$, $u_2 = -1487.2s$ and $u_3 = -1487.2s$. For each robot i, $-u_i$ is the expected time it would take it to complete its assigned region. This allocation



(a) (b)



(c)

Figure 3–1: Robot speed profiles for a coverage task in a square region

is optimal since all robots receive the same utility, the fastest robot is assigned more work, and swapping elements between the allocations would decrease the utilities of the robots.

Figure 3–3 shows a result of the allocation algorithm with 5 robots and a region defined by a non-convex polygon. As mentioned before, the algorithm will still find time-optimal solutions as long as the densities f_i are absolutely continuous with

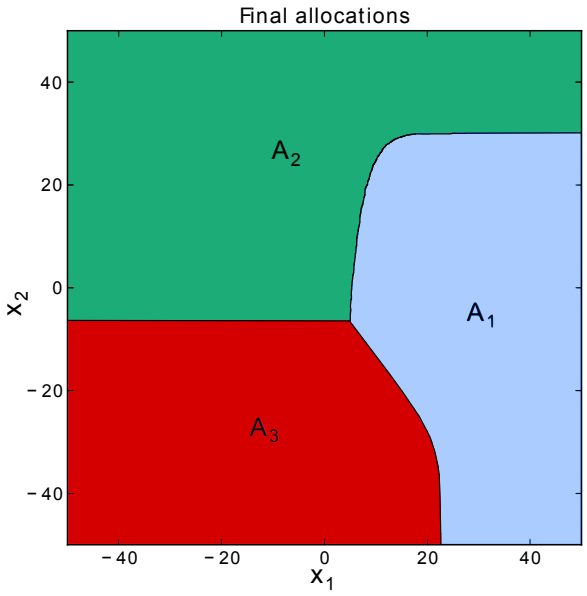


Figure 3–2: Final allocation of regions A_i to each robot $i \in R$

respect to each other. In a non-convex environment additional considerations on the allocations should be taken, such as connectedness and reachability.

It must be noted that this algorithm may not converge if $u_i(A) = u_j(A)$ for some $i, j \in R$ and a set A of measure greater than 0. In this case, the algorithm will oscillate between assigning A to i and j. Figure 3–4 shows how the error decreases non-monotonically, with the oscillations at the end demonstrating the aforementioned effect. One way of dealing with this is to assign every $x \in A$ uniformly at random to any robot i with $f_i(x) = \max_j f_i(x)$. If the task is defined in a metric space, then other properties of the allocations A_i can be used to resolve the conflicts such as connectedness and minimizing distances. In our coverage example, the robot position is an element of the same space as the task, therefore there exists a distance

Algorithm 1: Min-max time task subdivision

Inputs: A set $T \in \mathbb{R}^d$ which represents the task to be completed;
Speed profiles $s_{i,\max}$ for each robot $i \in R$;
A sequence of α_t that satisfies (3.12);
An error tolerance $\epsilon > 0$ and a maximum number of iterations $t_{\max}$.

Output: A function $K_{\max} : T \to R$ that encodes the allocation $A = \{A_1, \ldots, A_n\}$ which satisfies
$\bigcup_{i=1}^{n} A_i = T$,
$A_i \cap A_j = \emptyset \quad \forall i \neq j$,
and minimizes the maximum time spent by any robot;

$\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_n)$
$\lambda_i \leftarrow 1/n \quad \forall i \in R$
$t \leftarrow 1$, $\epsilon_t \leftarrow \infty$
$f_i(x) = -\frac{1}{s_{i,\max}(x)} \qquad \forall x \in T, i \in R$
while $t \leq t_{max} \wedge \epsilon_t \leq \epsilon$ **do**
    for each $x \in T$ **do**
        $K_{\max}(x) \leftarrow \arg\max_i \lambda_i f_i(x)$
        for each $i \in R$ **do**
            $$\mathbb{1}_i(x) \leftarrow \begin{cases} 1, & \text{if } K_{\max}(x) = i \\ 0, & \text{otherwise} \end{cases}$$
    $g_t \leftarrow \sum_{i=1}^{n} \int_T \mathbb{1}_i(x) \lambda_i f_i \mathrm{d}T$
    if $g_t \leq g_{best}$ **then**
        $g_{best} \leftarrow g_t$
        $K_{best} \leftarrow K_{\max}$
    for each $i \in R$ **do**
        $\gamma_i \leftarrow \int_T \mathbb{1}_i(x) f_i \mathrm{d}T$
    $\bar{\gamma} \leftarrow (\sum_{i=1}^{n} \gamma_i)/n$
    $\bar{\boldsymbol{\gamma}} = (\bar{\gamma}, \ldots, \bar{\gamma})_{1 \times n}$
    $\alpha \leftarrow \alpha_t$
    for each $i \in \{x \in R : \gamma_i^{(t)} > \bar{\gamma}\}$ **do**
        if $\alpha > \lambda_i^{(t)}/(\gamma_i^{(t)} - \bar{\gamma})$ **then**
            $\alpha \leftarrow \eta \lambda_i^{(t)}/(\gamma_i^{(t)} - \bar{\gamma})$
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \alpha(\boldsymbol{\gamma} - \bar{\boldsymbol{\gamma}})$
    $t \leftarrow t + 1$, $\epsilon_t \leftarrow (\max_{j \in R} u_j - \min_{j \in R} u_j)/\max_{j \in R} u_j$
return K_{best}

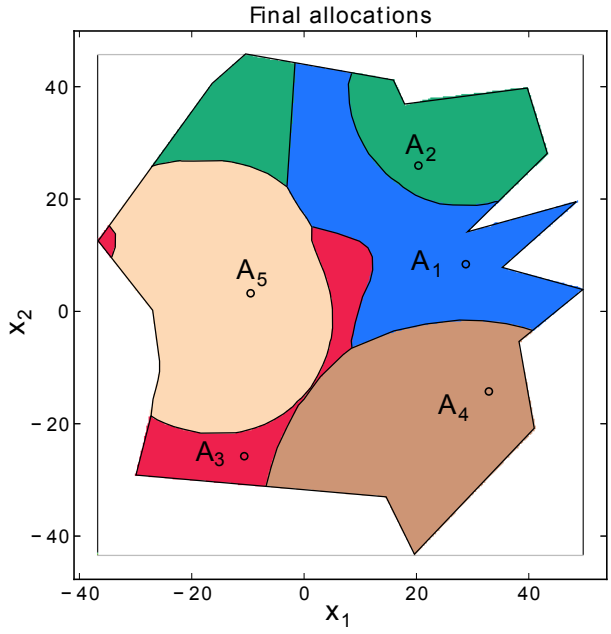


Figure 3–3: Final allocation of regions A_i to each robot $i \in R$ for an example with 5 robots.

function between the robot position $p_i \in \mathbb{R}^d$ and every point $x \in T$. Thus, for any conflicting set A we could replace every f_i with a

$$f_i^{'}(x) = f_i(x) - \kappa d_i(p_i, x)$$

where κ determines the influence of the distance factor on the solution. In such cases, the problem becomes a generalized version of finding a Voronoi tessellation of A. Figure 3–5 shows the case for three robots with speed profiles defined as

$$s_1(x) = \frac{2}{1 + e^{-40x_1+20}} \quad m^2/s$$

$$s_2(x) = 1 \quad m^2/s$$

$$s_3(x) = 1 \quad m^2/s$$

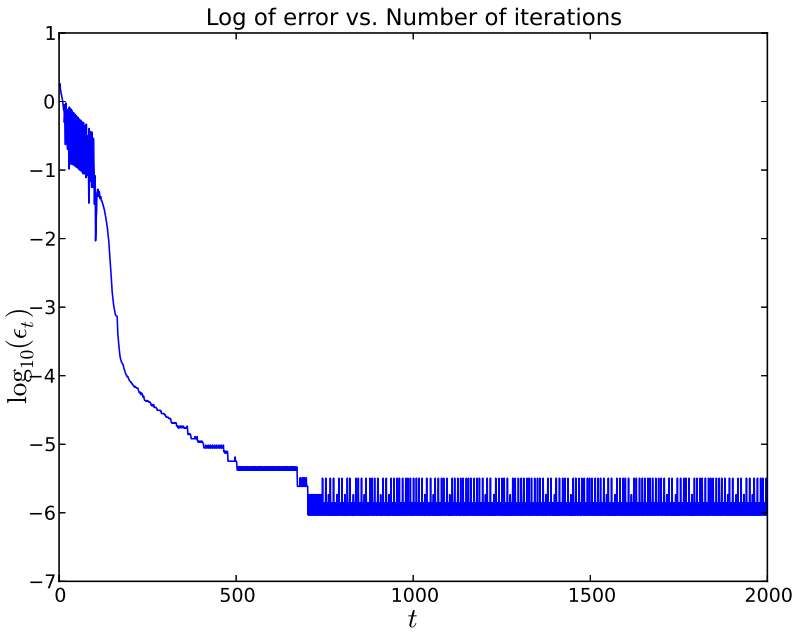
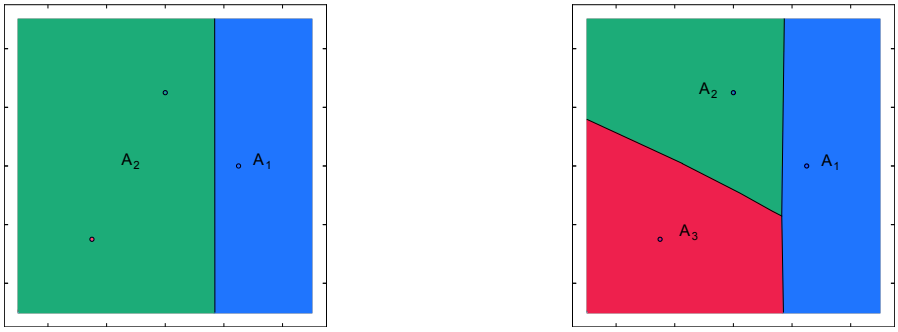


Figure 3–4: Logarithmic plot of error vs. number of iterations

The speed profiles were chosen to demonstrate the conflict that arises for having two robots with the same utility density functions. We compare the allocations made with the modified utility densities f_i', with $\kappa = 0$ and $\kappa = 5e - 6$. In the first case the final utilities are $u_1 = -1662.7$, $u_2 = -3337.5$ and $u_3 = 0$; the algorithm fails to split the task between the robots 2 and 3. By introducing the distance term the final utilities are $u_1 = -1667.1$, $u_2 = -1667.2$ and$u_3 = -1667.1$, balancing the load between the three robots.

3.3.1 Complexity analysis of algorithm 1

Both the time complexity and accuracy of Algorithm 1 depend on how T is discretized and the desired accuracy of the solution, ϵ. Let $|T|$ be the number of



(a) (b)

Figure 3–5: Example of introducing a distance term on the utility densities f_i (a) Allocation with no distance term (b) Allocation with distance term weighted by $\kappa = 5e-6$. The dots denote the positions of the robots.

elements into which T is discretized. Each iteration of the subgradient update executes $n(|T|+1) + O_{int}(n+1)$ operations, where O_{int} is the number of operation in the numerical integration method used. For example, if integration is done by using a midpoint or trapezoidal rule over the discretization of T, then $O_{int} = |T|$ and the run time of each iteration is $O(n|T| + n + |T|)$.

To obtain an estimate of the number of iterations until convergence we use the same analysis as in [38]. We need to determine distance the point $\boldsymbol{\lambda}^{(t)}$ has to traverse from the initial point to the optimal solution. Let $\boldsymbol{\lambda}^*$ be the optimal point. First, we note that $g(\boldsymbol{\lambda})$ is Lipschitz continuous since we assume every f_i is absolutely continuous. Therefore, the subgradient $\boldsymbol{\gamma}$ is bounded by some constant K_1 such that $\|\boldsymbol{\gamma}\| \leq K_1$. This bound is attained when at the optimal point, i.e. when $u_i = u^* \quad \forall i$. Thus, $\|\boldsymbol{\gamma}\| \leq K_1 = \sqrt{n\ u^{*2}}$. The distance between the starting point $\lambda_i = 1/n$, $\forall i \in R$ and the optimal point $\boldsymbol{\lambda}^*$ is also bounded by a constant K_2. Since

$D_{\boldsymbol{\lambda}}$ is the n-simplex, K_2 is the distance between the center of the n-simplex and one of its corners. Therefore,

$$\|\boldsymbol{\lambda}^{(1)} - \boldsymbol{\lambda}^*\|^2 \leq K_2^2 = \left(1 - \frac{1}{n}\right)^2 + \sum_{i=1}^{n} \left(\frac{1}{n}\right)^2 = 1 - \frac{1}{n}$$

These two bounds K_1 and K_2 will allow us to estimate the number of steps needed to converge up to an accuracy of ϵ. At the time step $t+1$ we have the following recursive definition,

$$\begin{aligned}
\|\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^*\|^2 =& \|\boldsymbol{\lambda}^{(t)} - \alpha_t \boldsymbol{\gamma}^{(t)} - \boldsymbol{\lambda}^*\|^2 \\
=& \|(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*) - \alpha_t \boldsymbol{\gamma}^{(i)}\|^2 \\
=& \|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*\|^2 - 2\alpha_t \boldsymbol{\gamma}^{(t)T}(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*) + \alpha_t^2 \|\boldsymbol{\gamma}^{(i)}\|^2 \\
\leq& \|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*\|^2 - 2\alpha_t (g(\boldsymbol{\lambda}^{(t)}) - g(\boldsymbol{\lambda}^*)) + \alpha_t^2 \|\boldsymbol{\gamma}^{(t)}\|^2 \\
\leq& \|\boldsymbol{\lambda}^{(1)} - \boldsymbol{\lambda}^*\|^2 - 2\sum_{i=1}^{t} \alpha_i (g(\boldsymbol{\lambda}^{(i)}) - g(\boldsymbol{\lambda}^*)) + \sum_{i=1}^{t} \alpha_i^2 \|\boldsymbol{\gamma}^{(i)}\|^2
\end{aligned}$$

where we have used the inequality 3.9 and replaced $\|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*\|^2$ recursively. Using our definition of K_2 and the fact that $\|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*\|^2 \geq 0$, we get the inequality,

$$2\sum_{i=1}^{t} \alpha_i (g(\boldsymbol{\lambda}^{(i)}) - g(\boldsymbol{\lambda}^*)) \leq K_2^2 + \sum_{i=1}^{t} \alpha_i^2 \|\boldsymbol{\gamma}^{(i)}\|^2$$

Since $g(\boldsymbol{\lambda}_{best}) - g(\boldsymbol{\lambda}^*) \leq g(\boldsymbol{\lambda}^{(i)}) - g(\boldsymbol{\lambda}^*) \quad \forall i$ and $\|\boldsymbol{\gamma}^{(i)}\| \leq K_1 \, \forall i$ the previous inequality can be rewritten as

$$g(\boldsymbol{\lambda}_{best}) - g(\boldsymbol{\lambda}^*) \leq \frac{K_2^2 + K_1^2 \sum_{i=1}^{t} \alpha_i^2}{2\sum_{i=1}^{t} \alpha_i} \tag{3.14}$$

The right hand side of this inequality is minimized when

$$\alpha_i = \frac{K_2}{K_1\sqrt{t}} \qquad \forall i$$

providing a bound on the error. Replacing this value in 3.14 and setting $g(\boldsymbol{\lambda}_{best}) - g(\boldsymbol{\lambda}^*) = \epsilon$ yields

$$\epsilon \leq \frac{K_2^2 + K_1^2 t \left(\frac{K_2}{K_1\sqrt{t}}\right)^2}{2t\frac{K_2}{K_1\sqrt{t}}} = \frac{K_1 K_2}{\sqrt{t}}$$

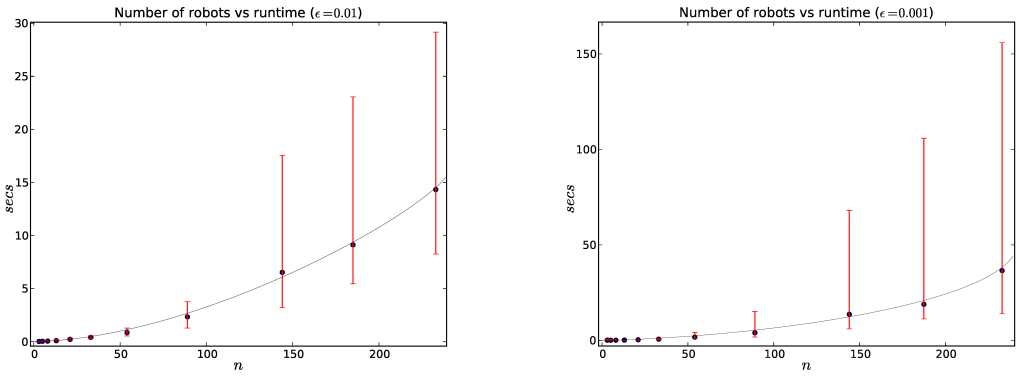
which means that the algorithm must be run for at least

$$\left(\frac{K_1 K_2}{\epsilon}\right)^2 \approx n\left(\frac{u^*}{\epsilon}\right)^2$$

steps to achieve the desired accuracy. Combining this number with the run time of each iteration, we have that the complexity of the algorithm is $O(n\left(\frac{u^*}{\epsilon}\right)^2 (n(|T| + 1) + O_{int}(n+1))$. For the case when the trapezoidal rule is used, the complexity is $O((u^* n)^2 |T|/\epsilon)$. Figure 3–6 shows the increase in running time with increasing number of robots.

3.3.2 Balancing energy costs

As mentioned before, another way to ensure load balancing is to take the max-sum problem and introduce energy cost constraints. Let $C_i(A)$ denote the cost for robot i to perform the subtask $A \in T$. We assume that each $C_i(A) > 0, \quad \forall A \subseteq T$ is a countably additive measure over the subsets of T. Let $c_i(x)$ be the density of C_i over T. Let $\mathcal{E}_i < C_i(T)$ be the energy capacity of robot i. Given the energy



(a) (b)

Figure 3–6: Plot of the increase in runtime (in seconds) against the number of robots. The data points correspond to the average of 20 runs of the algorithm with 2, 3, 5, 8, 13, 21, 33, 54, 89, 144, 180 and 233 robots, with $\epsilon = 0.01$ and $\epsilon = 0.001$. The dots are the mean running time in seconds, the bars show the minimum and maximum runtime for each value of n

constraint, we want to solve the following optimization problem,

$$\begin{aligned} \max_{A} \quad & \sum_{i=1}^{n} \int_{A_i} f_i \mathrm{d}T \\ \text{subject to} \quad & A_i \cap A_j = \emptyset \qquad \forall i \in R, j \in R \\ & \bigcup_{i=1}^{n} A_i = T \\ & \int_{A_i} c_i(x) \mathrm{d}T \leq \mathcal{E}_i \end{aligned} \tag{3.15}$$

If the robots do not have enough energy to complete the whole task then the problem becomes unfeasible. We will first assume that the combined energy capacities of the robots are enough to complete the task. By using the indicator function $\mathbb{1}_i$, this can

be rewritten as

$$\begin{aligned} \max_{\mathbb{1}_1,\ldots,\mathbb{1}_n} \quad & \sum_{i=1}^{n} \int_T \mathbb{1}_i(x) f_i(x) \mathrm{d}T \\ \text{subject to} \quad & \int_T \mathbb{1}_i(x) c_i(x) \mathrm{d}T \leq \mathcal{E}_i \qquad \forall i \in R \\ & \sum_{i=1}^{n} \mathbb{1}_i(x) = 1 \qquad \forall x \in T \\ & \mathbb{1}_i(x) \in \{0,1\} \qquad \forall i \in R, x \in T \end{aligned} \tag{3.16}$$

By a similar procedure as in equation 3.5, we can formulate the dual of 3.15 as

$$\begin{aligned} \min_{\lambda_1,\ldots,\lambda_n,\Lambda} \quad & \sum_{i=1}^{n} \lambda_i \mathcal{E}_i + \int_T \Lambda(x) \mathrm{d}T \\ \text{subject to} \quad & \Lambda(x) \geq f_i(x) - \lambda_i c_i(x) \quad \forall i \in R, x \in T \end{aligned} \tag{3.17}$$

The function $\Lambda(x)$ depends on the λ variables and the optimal solution Λ^* should satisfy $\Lambda^*(x) = f_i(x) - \lambda_i^* c_i(x)$. Consequently, we can rewrite the dual problem as

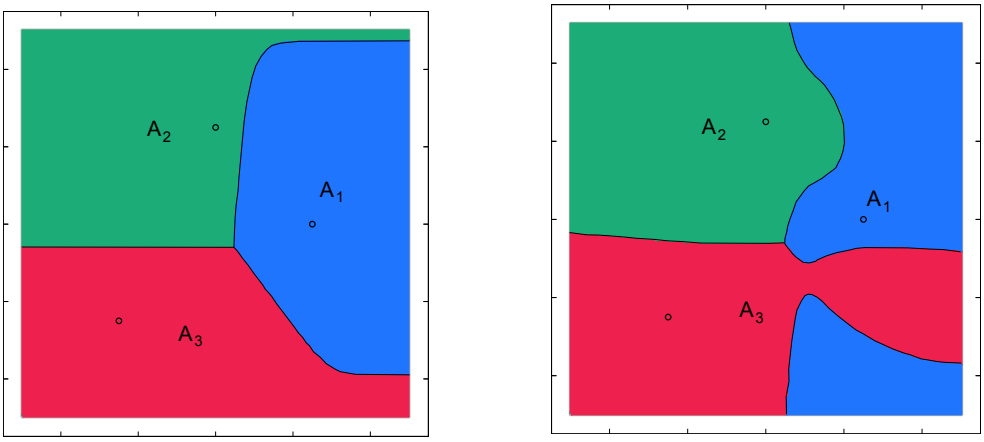
$$\min_{\lambda_1,\ldots,\lambda_n} \quad \sum_{i=1}^{n} \lambda_i \mathcal{E}_i + \int_T \max_i \left\{ f_i(x) - \lambda_i c_i(x) \right\} \mathrm{d}T \tag{3.18}$$

By following the same procedure of the previous sections, it is simple to show that a subgradient for this algorithm is given by

$$\gamma_i^{(t)} = \mathcal{E}_i - \int_T \mathbb{1}_{i,max}^{(t)}(x) c_i(x) \mathrm{d}T \tag{3.19}$$

where $\mathbb{1}_{i,max}$ is given by

$$\mathbb{1}_{i,max}^{(t)}(x) = \begin{cases} 1, & \text{if } f_i(x) - \lambda_i^{(t)} c_i(x) \geq f_j(x) - \lambda_j^{(t)} c_j(x) \qquad \forall j \neq i \\ 0, & \text{otherwise} \end{cases}$$



(a) (b)

Figure 3–7: Energy balanced allocations for the example in section 3.2, using area (a) and distance (b) as the energy cost functions.

The first term in Eq. 3.19 is the energy capacity, the second term is the cost associated to the current allocation for robot i. Intuitively, the λ_i variables represent how much importance will be given to the cost functions in order to satisfy the energy constraints. Starting with $\lambda_i = 0$ for all i, the initial allocation will be the max-sum allocation; i.e. with nothing allocated, no robot violates its energy constraint, thus the objective function has the energy terms equal to 0. If any robot i violates the energy constraint then $\gamma_i < 0$, thereby increasing its corresponding λ_i on the next time step. On the other hand, if robot i does not violate the energy constraint, meaning it still has some energy to spend, then $\gamma_i > 0$, reducing the value of its cost term $\lambda_i c_i(x)$ on the next time step. This has the effect of changing the sizes of the allocations of the robots to minimize the difference $\left|\mathcal{E}_i - \int_T \mathbb{1}^{(t)}_{i,max}(x)c_i(x)\mathrm{d}T\right|$.

Figure 3–7 shows two examples of energy balanced allocations for the example setting of section 3.2. In one of the examples the cost functions are $c_i(x) = 1$ for all x; i.e. the cost is the allocated area. In the other example the cost is the distance from the point x to the location of the robot. The algorithm for computing the energy balanced partitions is the same as Algorithm 1, except for the computation of γ_i and $\mathbb{1}_{i,max}^{(t)}(x)$, and the stopping criterion. In this version of the task subdivision problem, the optimal solutions are refinements of the max-sum solution in which there is a restriction on how much work we can allocate to each robot. Depending on the energy capacities and costs, not every robot will be allocated a subtask. Subtasks are greedily allocated to the fastest robots, and the load is subsequently reduced for robots that violate their energy capacity constraint. Therefore the subgradient search algorithm should stop when $\gamma_i = 0$ for the robots with $A_i \neq \emptyset$; i.e. the robots that have been allocated a subtask.

3.4 Summary

In this chapter we've formulated the task subdivision problem for continuous tasks as a fair division problem. We provided an algorithm for task subdivision with implicit load balancing and showed some convergence properties, inherited from the subgradient method used for performing the subdivision. We also showed how to include energy balancing constraints explicitly and still use the same algorithm to find the corresponding task subdivision. In both cases, the subdivision is performed by a centralized algorithm, for which every robot should communicate its density function f_i and possibly its energy cost and capacities, c_i and $\mathcal{E}_i$. In the next section we provide a first approach for a decentralized algorithm for task subdivision.

CHAPTER 4
Decentralized task subdivision

When performing tasks with multiple robots, sometimes it is desirable to decentralize the computation of the task subdivision. We will concentrate on the max-min problem of the previous section. One approach is that of distributing the computation of the subgradient algorithm by letting each robot receive incremental updates on the components of the subgradient vector from other robots [39]. If updates to the subgradient components are received uniformly at random, each γ_i will on average be updated the same number of times, and the local solutions will converge to the centralized optimal solution. Unfortunately, in our setting it is not possible to guarantee that updates on subgradient components will occur uniformly at random, unless every robot can communicate constantly with the other robots. To deal with this, our first approach is to let the robots move randomly inside their allocated partitions and let them refine their local partitioning with updates from other robots.

4.1 Decentralized algorithm

We define a degree of heterogeneity between two robots, measured by the sum of squares difference of their utility density functions f_i, f_j, as

$$\mathrm{SSD}_{ij} = \int_T \left(f_i(x) - f_j(x)\right)^2 \mathrm{d}x \tag{4.1}$$

This number gives an idea of how similar two robots are. Every robot has a communications range r_i. Our design decision is that initially every robot has knowledge

only of its own density f_i and assumes every other to be uniform over T. The value of these assumed uniform densities should be the same for all robots $j \neq i$ and equal to some small number. This should make robot i allocate most of T to itself, from its local information. Each robot will compute its own subgradient vector with its local information and broadcast its utility density f_i when requested by other robots in range.,The robot i will also request the corresponding f_j's of the other robots in range and update its estimated $\boldsymbol{\lambda}_i$, which in turn updates its local allocation $\boldsymbol{A}_i = \{A_{i1}, \ldots, A_{in}\}$. Algorithm 2 describes this procedure for every robot.

The idea is to drive each robot to its current locally assigned subtask A_{ii}. If two robots i, j have a small value for $SSD_{i,j}$, then their allocations A_{ii} and A_{jj} will be similar, eventually driving themselves within communications range. Otherwise the distance $SSD_{i,j}$ will be large and both robots will have non-conflicting allocations. This ignores the fact that offsets between the robots timings results in the possibility of repeated work, by having one robot follow another, but never getting within communications range. To avoid this, for the following sections, every robot will drive itself towards k locations randomly picked within their allocations A_i. Other approaches such as letting robot i follow the centroid of A_i or performing systematic coverage over the connected pieces of A_i could be considered, but the random tour over k locations provides a baseline of performance for our algorithm that is general enough. To see this, assume there are n locations to visit in a grid over a conflicting allocation between robot i and j. Assume it takes a single time step to transition between neighboring locations. Then the time it will take the robots to meet each other is the time for two random walks to reach the same state. Assume that the

Algorithm 2: Decentralized task subdivision (executed by every robot i)

Inputs: A local utility density function f_i;
The number of robots n;
A sequence of α_t that satisfies (3.12);
An error tolerance $\epsilon > 0$ and a maximum number of iterations $t_{\max}$.

Output: A function $K_{i,\max} : T \to R$ that encodes the allocation
$\boldsymbol{A}_i = \{A_{i1}, \ldots, A_{in}\}$
which satisfies
$\bigcup_{i=1}^{n} A_i = T$,
$A_i \cap A_j = \emptyset \quad \forall i \neq j$,
and minimizes the maximum time spent by any robot;

Set every local f_j for $j \neq i$ to some initial estimate (i.e.$f_j(x) = \text{constant}$)
$\lambda_i \leftarrow 1/n \quad \forall i \in R$
Compute an initial subgradient vector $\boldsymbol{\gamma}$ from prior information
Initialize the set of encountered robot K_i as $\{i\}$
while **true do**
    Move to a previously unvisited location inside A_i
    if robot is within communications range of **then**
        Request and update f_j
        $K_i \leftarrow K_i \cup j$
    Compute the local $K_{\max}$, $\mathbb{1}_i(x)$ with the information from the set of encountered robots K_i
    Update $\boldsymbol{\gamma}$
    Listen for other robots in range
    Broadcast f_i if requested
return K_{best}

grid is a rectangular region that covers the union of the conflicting allocations for robots i and j. By a result of Aldous [40] the meeting time of two such random walks is half the cover time of a single random walk. And as shown by Aldous [41] and Zuckerman [42], this cover time is $\Omega n \log^2 n$. Therefore the two robots will meet in a finite number of time steps, depending on the size of the conflicting region.

Every robot $i \in R$ initially assumes it is the best suited robot for all parts of the task T. At any time, when no new information is available, a robot might have assigned to itself a non-conflicting part of the task, which means that its $SSD_{i,j}$ is large for any $j \in R$, or assign itself a part of the task in which it will encounter other robots with similar preferences. In the latter case, new information will be available to refine task distribution.

We assume that the robots are completely heterogeneous, i.e., there are no two robots with equal densities $f_i = f_j$. At the beginning, every robot will assign the whole task T to itself. The choice of the number of locations k to visit, determines the chances that any two robots will meet. This part is similar to the *rendezvous* problem [2, 43, 44]; except the robots are not actively trying to meet each other. Instead, robot i constructs the set of k locations by picking them uniformly at random from the partition A_i. If robot i fails to meet any other robot, it tries again. Since at this time every robot assigns to itself the whole task T, the probability that any robot i meets another robot j at the first trial is some positive number that depends on the minimum communication radius among robots r. This probability is non-zero as long as r is not much smaller than the size of T. If any two robots i and j come within communications range and exchange their utility density functions, then their

local partitioning will be the same and both robots will move away from each other, as their allocated subtasks will be disjoint– from solving the max-min problem with the locally available information.

In general each robot time a pair of robots i and j exchange information, their allocated subtasks will be disjoint, And the task will be split in half between the pair. As information from new robots is acquired, each robot will repartition the task among its local group of m robots. If a robot ends without meeting all n robots, it will end up covering some partition for which it is the best suited robot.

An example of a run of Algorithm 2 for 6 robots over a square region is shown in Figures 4–2. The speed profiles used for this example are shown in Figure 4–1. Even though none of the robots has complete information to compute globally optimal partitions, e.g. robot 6, their locally assigned partitions approximate the ones of the centralized solution. The paths traversed by the robots, figure 4–3, illustrate how each robot ends covering a region that roughly corresponds to the allocation computed with the centralized algorithm, shown in Figure 4–4.

A remaining issue to study is the effect of varying the radius r on the running time until finding a partition, and the number of trials until obtaining information form a new robot. We hypothesize that increasing the radius will bring the number of iterations closer to that of the centralized approach. It must be noted that the centralized algorithm, in general, requires less computation and finds partitions with smaller values for the objective function g. To see this it is enough to consider that each robot is running the centralized algorithm with local information, and updates to the local information come after a possibly long wait time. Nevertheless, the

motivation for a decentralized algorithm is the added robustness to partial failures and the fact that robots do work while searching for partitions.

4.2 Clustering similar robots

Another approach is to let the robots form coalitions to perform task subdivision. To do this, we first cluster the robots into k groups. This clustering is based on the $SSD_{i,j}$ similarities and corresponds to finding k consecutive minimum cuts of the graph $G(V, E)$, where the vertices are the robots $i \in R$ and the edges have cost $SSD_{i,j}$. The k clusters are found by finding the minimum cut of the biggest connected component of G; i.e. first split G into$\{G_1, G_2\}_1$, then split the biggest between the two connected components to obtain $\{G_1, G_2, G_3\}_2$, and so on. Once the k clusters have been found, the robots utility densities are combined as

$$F_k(x) = \sum_{i=0}^{n_k} \frac{n}{n_k} \mathbb{1}_{max}(x) f_i(x), \qquad \forall x \in T$$

This is done to ensure that a group of n_k robots receives a part of the task that is proportional to n_k. This is similar to the weighting strategy used in the analysis of cooperative fair division games [37]. After the clustering is done, the machine that performed the clustering proceeds to perform task subdivision among the k clusters. Once this is done, each task subdivision is either done by recursive clustering or by the decentralized approach of the previous section.

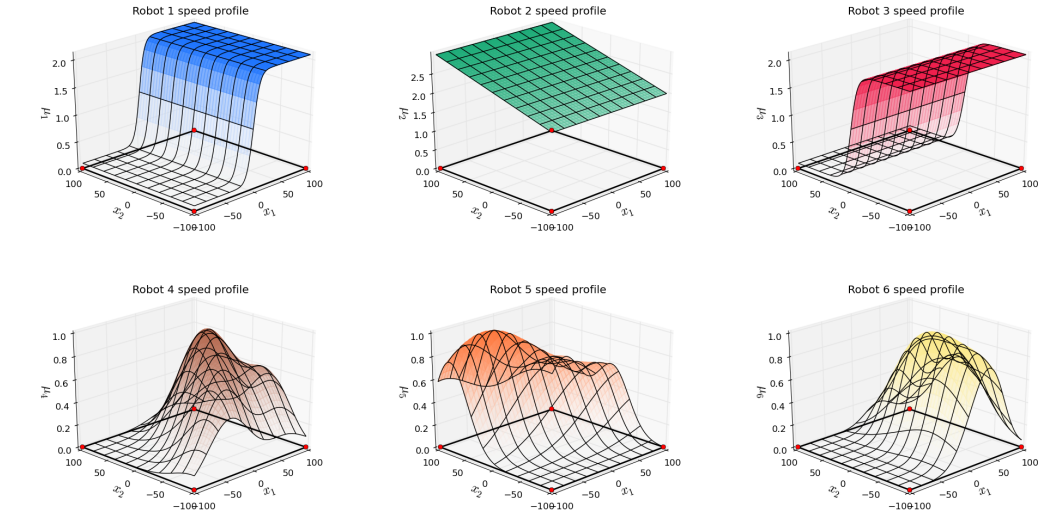
This algorithm will no longer produce an optimal partition, but rather it will approximate one as long as the edges that were cut had values close to 0. Otherwise a robot that could belong to multiple clusters will only be allocated a fraction of the optimal partition.

4.3 Summary

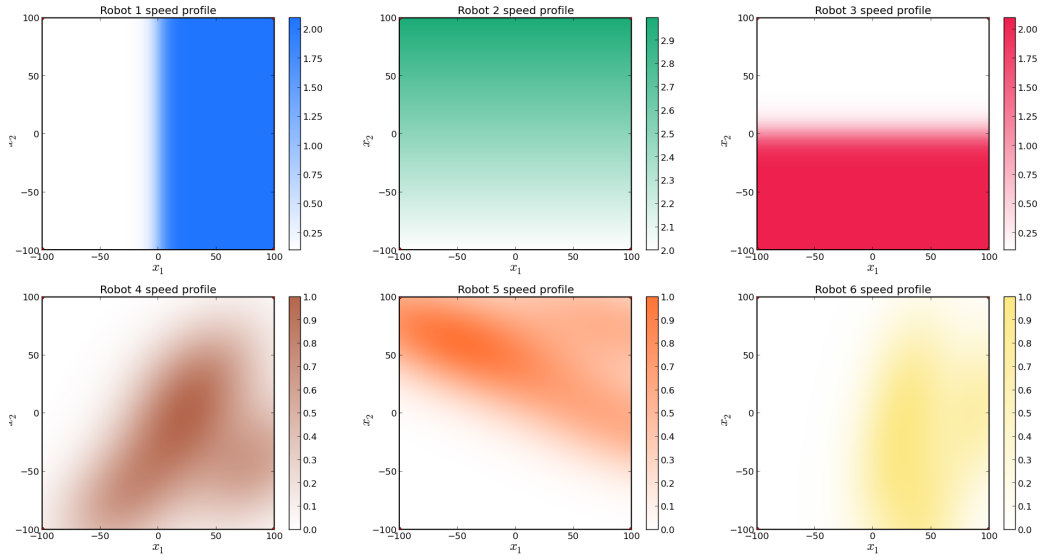
In this chapter we presented ideas for decentralized and partially centralized approaches for task subdivision with a group of n robots performing a coverage task. We discussed the reasons why such algorithm might work and the algorithms through simulation. As mentioned before, the centralized algorithm might provide the best results; but this is under the assumption of full communication between the robots and a central "master" processor. The decentralized algorithm does not make that assumption, since every robot computes a local task subdivision, which is refined as the robots obtain new information.

Although we only mentioned the case of obtaining new information about other robots, the decentralized algorithm should also work in cases where the robots have no previous knowledge of their utility density f_i. Updating the function f_i only changes the search direction, with an impact on the convergence time, as the allocations computed with previous information are a valid starting point. To avoid falling in a local minima, the robots reset the step size of their sub gradient search to a bigger value.

We also presented an approach for task subdivision through recursive clustering. This approach allows us to off load the computation from single robots to intermediate, possibly more powerful processors. The price to pay for the recursive clustering is not only computation time but also the possibility of having a robot whose max-min allocation within the cluster might only a fraction of the optimal allocation.

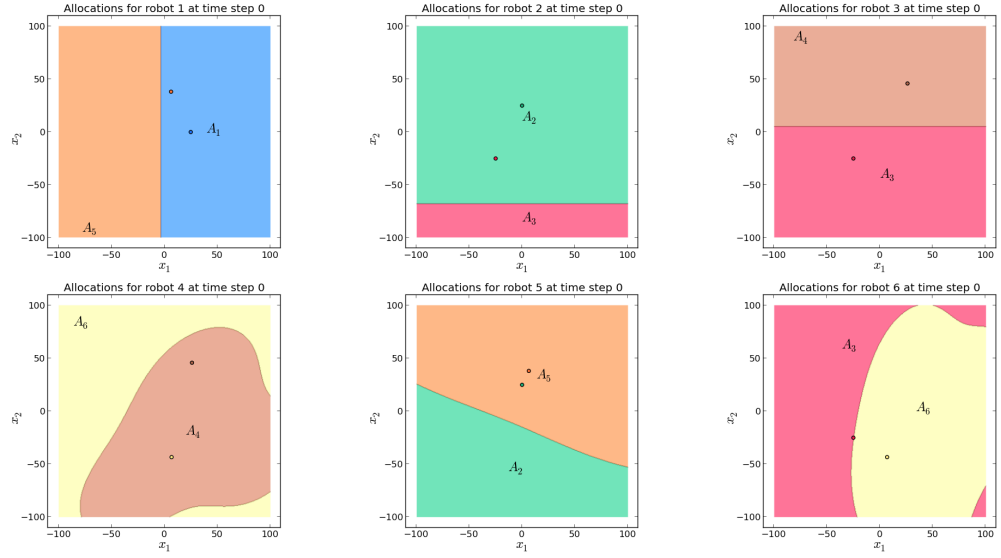


(a)

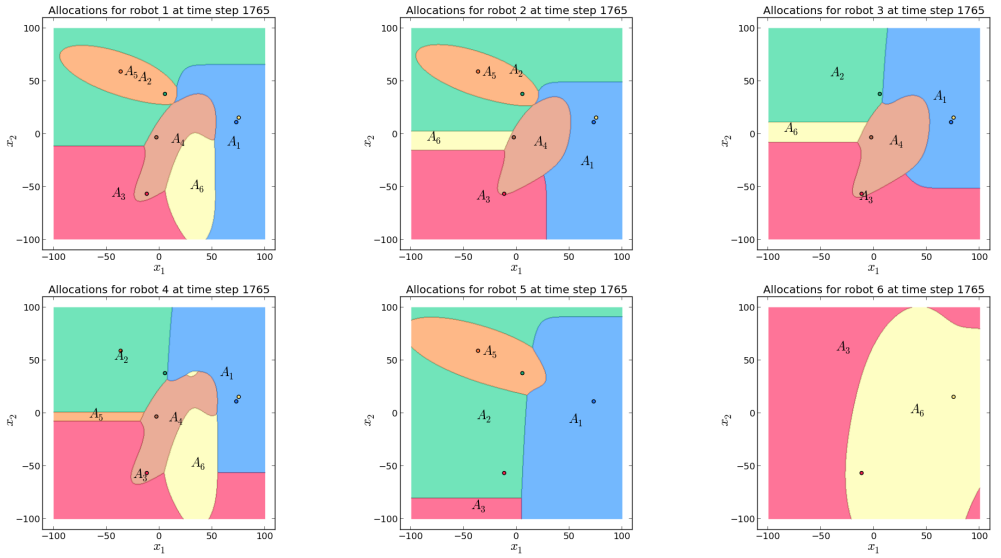


(b)

Figure 4–1: Speed profiles used for the example run of Algorithm 2. (a) shows a 3d view of the speed profiles. (b) show the same speed profiles in a density plot; darker means faster.



(a)



(b)

Figure 4–2: Allocations computed locally by each robot. The allocations shown correspond to the ones at time 0 (a), and after 1765 time steps (b). Robot 6 never exchanges information with other robots

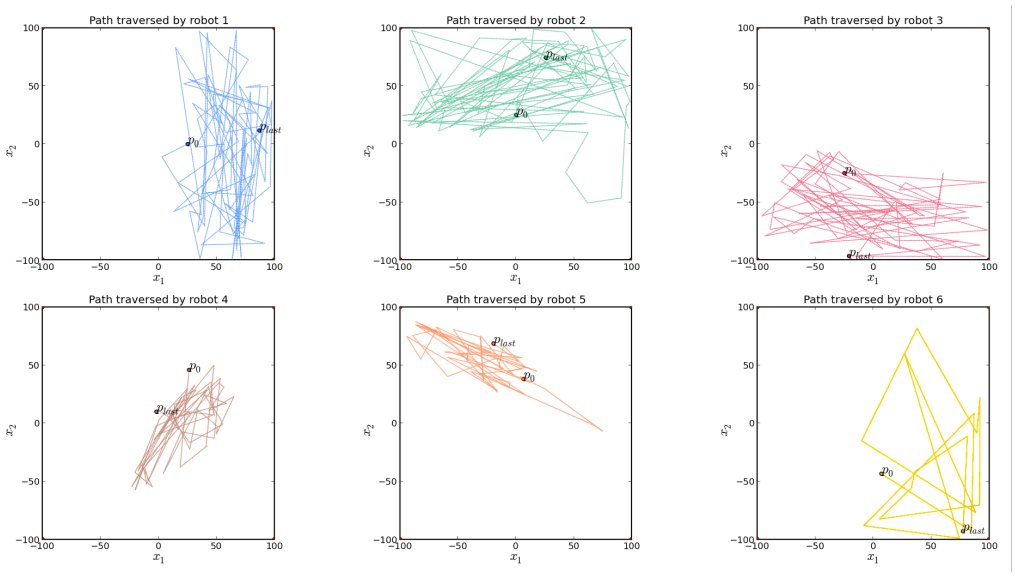


Figure 4–3: Paths traversed by the robots during the execution of Algorithm 2.

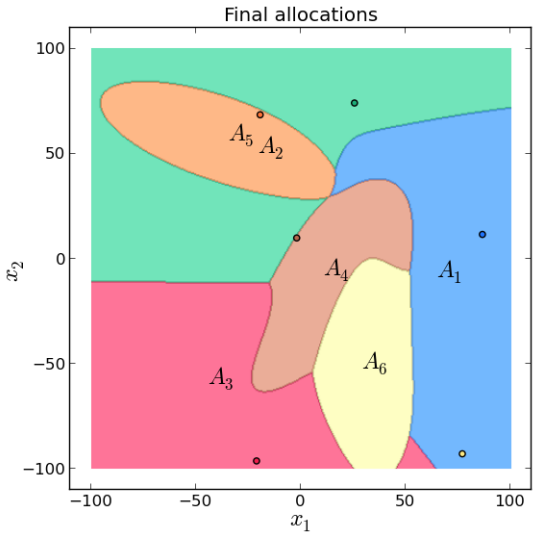


Figure 4–4: Allocations obtained by executing Algorithm 1, which is centralized..

CHAPTER 5
CONCLUSION

In this work we presented an approach for subdividing and allocating a single global task between multiple heterogeneous robots. We have determined the computational complexity of the approach and illustrated its effectiveness. This approach deals with the heterogeneity of the robot team by letting the robots define a preference function over the task space. In the specific case of terrain coverage, we have illustrated how our approach can be used to find a suitable time-optimal task allocation that corresponds, as well, to a spatial decomposition. We suspect this same approach should be easily generalizable to $\mathbb{R}^n$, as long as integrals are computed efficiently, e.g., with a Monte Carlo algorithm. Such subdivision optimizes the time spent by any robot on its own allocation. These time estimates are useful for selecting meeting times and locations in a multi robot rendezvous problem [43]. Although the algorithm proposed in this work is able to find partitions of arbitrary speed profiles efficiently, there is no guarantee that the partition is the best one for real robots with non-holonomic constraints. In such cases, the assumption of additive utility function will not hold; i.e., depending on the motion model of the robots, some shapes of allocations will be preferred.

The algorithms presented in this work need to be tested with real robots to provide more insights on better strategies for task subdivision and for validating if the fair division approach is good enough. These experiments should give us an idea

of how to better relate robot capabilities and motion constraints to the preference scoring functions.

Finally, the algorithm needs to be tested on a probabilistic setting in which the robots have no previous information on how their capabilities relate to task performance. In this case, the robots will need to measure their performance from their environment and from proprioception; measurements that will be noisy.

5.1 Future Work

There are multiple avenues of our interest for future work. One is the performance guarantees in the case of incomplete information. This problem arises in multi-robot systems as no communication channel is completely reliable. Thus it is of our interest to see what are the effects of incomplete, out of date and possibly wrong information on the convergence time for the allocations.

We are also interested in the time varying version of our problem. Certain events might produce changes in the environment or robot abilities, rendering previous allocations invalid. A similar situation occurs when the robots have no previous information: as robots explore the task space, new information brings the need for reallocation of the subtasks.

Another avenue for research is to introduce transitive communication. In our work we only considered 1-hop communication which might make robots converge to regions in which they are locally the best suited robot, with respect to neighboring robots; this allocation may be far from optimal since the robot does not have information from regions far from its current allocation.

A particular topic of interest is that of combining task allocation and rendezvous [43]. After a first allocation, we studied the problem of distributed and iterated task subdivision by letting the robots move randomly inside their allocated region. This was done to guarantee that the robots would meet in finite time. So the question is if we can do better by having the robots move systematically within their allocated regions and use a smarter strategy to find each other to exchange information.

References

[1] Gregory Dudek, Michael R. M. Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for multi-agent robotics. *AUTONOMOUS ROBOTS*, 3:375–397, 1996.

[2] Nicholas Roy and Gregory Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11:117–136, 2000.

[3] Brennan Peter Sellner, Frederik Heger, Laura Hiatt, Reid Simmons, and Sanjiv Singh. Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE - Special Issue on Multi-Robot Systems*, 94(7):1425 – 1444, July 2006.

[4] M. Ani Hsieh, Anthony Cowley, James F. Keller, Luiz Chaimowicz, Ben Grocholsky, Vijay Kumar, Camillo J. Taylor, Yoichiro Endo, Ronald C. Arkin, Boyoon Jung, Denis F. Wolf, Gaurav S. Sukhatme, and Douglas C. MacKenzie. Adaptive teams of autonomous aerial and ground robots for situational awareness: Field reports. *J. Field Robot.*, 24(11-12):991–1014, November 2007.

[5] Brian P. Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The Intl. J. of Robotics Research*, 23(9):939–954, sep 2004.

[6] N. Atay and B. Bayazit. Emergent task allocation for mobile robots. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[7] Robert Michael Zlot and Anthony (Tony) Stentz. Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research, Special Issue on the 4th International Conference on Field and Service Robotics*, 25(1):73–101, January 2006.

[8] Lantao Liu and Dylan Shell. Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[9] Harold W. Kuhn. The hungarian method for the assignment problem. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 29–47. Springer Berlin Heidelberg, 2010.

[10] Hannah Bast and Susan Hert. The area partitioning problem. In David Bremner, editor, *Proceedings of the 12th Annual Canadian Conference on Computational Geometry (CCCG-00)*, pages 163–171, Fredericton, New Brunswick, Canada, August 2000. University of New Brunswick.

[11] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Distributed algorithms for environment partitioning in mobile robotic networks. *Automatic Control, IEEE Transactions on*, 56(8):1834 –1848, aug. 2011.

[12] John Gunnar Carlsson. Dividing a territory among several vehicles. *INFORMS Journal on Computing*, 2011.

[13] Jörg Kienzle, Clark Verbrugge, Bettina Kemme, Alexandre Denault, and Michael Hawker. Mammoth: a massively multiplayer game research framework. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, pages 308–315, New York, NY, USA, 2009. ACM.

[14] L. E. Dubins and E. H. Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68(1):pp. 1–17, 1961.

[15] Marco Dall' Aglio. The dubins-spanier optimization problem in fair division theory. *J. Comput. Appl. Math.*, 130(1-2):17–40, May 2001.

[16] J. Cortes. Coverage optimization and spatial load balancing by robotic sensor networks. *Automatic Control, IEEE Transactions on*, 55(3):749 –754, march 2010.

[17] H. Sayyaadi and M. Moarref. A distributed algorithm for proportional task allocation in networks of mobile agents. *Automatic Control, IEEE Transactions on*, 56(2):405 –410, feb. 2011.

[18] Subhrajit Bhattacharya, Nathan Michael, and Vijay Kumar. Distributed coverage and exploration in unknown non-convex environments. In *Proceedings of 10th International Symposium on Distributed Autonomous Robotics Systems*. Springer, 1-3 Nov 2010.

[19] 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 *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4982 –4989, may 2010.

[20] J. Cortes. Coverage optimization and spatial load balancing by robotic sensor networks. *Automatic Control, IEEE Transactions on*, 55(3):749 –754, march 2010.

[21] L. Pimenta, V. Kumar, R.C. Mesquita, and G. Pereira. Sensing and coverage for a network of heterogeneous robots. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3947 –3952, dec. 2008.

[22] K. R. Guruprasad. Multi-agent search using sensors with heterogeneous capabilities. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: doctoral mentoring program*, AAMAS '08, pages 1734–1735, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[23] H. S. Stone. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Trans. Softw. Eng.*, 3(1):85–93, January 1977.

[24] Lantao Liu and Dylan Shell. Large-scale multi-robot task allocation via dynamic partitioning and distribution. *Autonomous Robots*, pages 1–17, 2012. 10.1007/s10514-012-9303-2.

[25] B.P. Gerkey and M.J. Mataric. Sold!: auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on*, 18(5):758 – 768, oct 2002.

[26] L.E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220 –240, apr 1998.

[27] Barry Brian Werger and Maja J. Mataric. Broadcast of local eligibility: behavior-based control for strongly cooperative robot teams. In *Proceedings of the fourth international conference on Autonomous agents*, AGENTS '00, pages 21–22, New York, NY, USA, 2000. ACM.

[28] Rodney A. Brooks. A robust layered control system for a mobile robot. Technical report, Cambridge, MA, USA, 1985.

[29] Fang Tang and Lynne E. Parker. Asymtre: Automated synthesis of multi-robot task solutions through software reconfiguration. In *In Proceedings of IEEE International Conference on Robotics and Automation*, pages 1513–1520, 2005.

[30] C. Rossi, L. Aldama, and A. Barrientos. Simultaneous task subdivision and allocation for teams of heterogeneous robots. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 946 –951, may 2009.

[31] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *Robotics and Automation, IEEE Transactions on*, 20(2):243 – 255, april 2004.

[32] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, December 1999.

[33] M. Schwager, M. P. Vitus, D. Rus, and C. J. Tomlin. Robust adaptive coverage for robotic sensor networks. In *Proceedings of the International Symposium on Robotics Research (ISRR 11)*, August 2011.

[34] Ishay Kamon, Elon Rimon, and Ehud Rivlin. Tangentbug: A range-sensor-based navigation algorithm. *The International Journal of Robotics Research*, 17:934–953, 1998.

[35] John Gunnar Carlsson. Dividing a territory among several facilities. Working paper (available at `http://menet.umn.edu/~jgc`), University of Minnesota, Twin Cities, 2011.

[36] Jack M. Robertson and William A. Webb. *Cake-cutting algorithms - be fair if you can.* A K Peters, 1998.

[37] Dall'Aglio Marco and Camilla Di Luca. Finding maxmin allocations in cooperative and competitive fair division. Working paper (available at `http://arxiv.org/pdf/1110.4241.pdf`), Fondazione Eni Enrico Mattei, 2011.

[38] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. Working paper (available at `http://www.stanford.edu/class/ee392o/subgradmethod.pdf`), Stanford University, 2003.

[39] A. Nedić, D.P. Bertsekas, and V.S. Borkar. Distributed asynchronous incremental subgradient methods. In Yair Censor Dan Butnariu and Simeon Reich, editors, *Inherently Parallel Algorithms in Feasibility and Optimization and their*

Applications, volume 8 of *Studies in Computational Mathematics*, pages 381 – 407. Elsevier, 2001.

[40] David J. Aldous. Meeting times for independent markov chains. *Stochastic Processes and their Applications*, 38(2):185 – 193, 1991.

[41] DavidJ. Aldous. Lower bounds for covering times for reversible markov chains and random walks on graphs. *Journal of Theoretical Probability*, 2:91–100, 1989.

[42] D. Zuckerman. A technique for lower bounding the cover time. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC '90, pages 254–259, New York, NY, USA, 1990. ACM.

[43] Malika Meghjani and Gregory Dudek. Combining multi-robot exploration and rendezvous. In *Proceedings of the 2011 Canadian Conference on Computer and Robot Vision*, CRV '11, pages 80–85, Washington, DC, USA, 2011. IEEE Computer Society.

[44] Steve Alpern. The rendezvous search problem. *SIAM J. Control Optim.*, 33(3):673–683, May 1995.