SPATIO-TEMPORAL MULTI-ROBOT ROUTING

A Thesis
Presented to
The Academic Faculty

by

Smriti Chopra

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2015

SPATIO-TEMPORAL MULTI-ROBOT ROUTING

Approved by:

Professor Magnus Egerstedt, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Spiridon Reveliotis
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Fumin Zhang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Tucker Balch
College of Computing
Georgia Institute of Technology

Professor Ayanna Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 19 March 2015

To my mum and dad.

jai mata di.

ACKNOWLEDGEMENTS

First off, I want to thank Magnus Egerstedt for being the best advisor, teacher and mentor one could ask for. It was at his hands that I learned the ropes of research, and the art of merging excitement and creativity with technical precision. I loved every harrowing minute of it, and will forever be grateful to him for letting me barge my way into the GRITS lab.

Which brings me to its members, my lab mates who were a massive support to me throughout the four years I spent here. I would especially like to thank JP, Yancy, Thiagarajan and Sung for being my rocks, patiently suffering through conversations on my innumerable research "crises", turning never-ending demo days into rollicking fun, and holding down the fort at all those times that I messed up. I am also grateful to Matt Rice, for being by my side (instead of bolting in the opposite direction), and powering through the research whirlwind that we experienced this past year.

Finally I would like to thank a handful of very special people, without whom I would not have survived even a month in the US. Suma, for being my family away from home, Hassan for being my begrudging counsellor away from home, Gaurav Mama and Pooja for being my trendy guardians away from home, and Mum, Dad, Sakshi, Siddharth, Shravan and Chutters, for being my everything else.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

In this dissertation, we analyze spatio-temporal routing under various constraints *specific to* multi-robot applications. Spatio-temporal routing requires multiple robots to visit spatial locations at specified time instants, while optimizing certain criteria like the total distance traveled, or the total energy consumed. Such a spatio-temporal concept is intuitively demonstrable through *music* (e.g. a musician routes multiple fingers to play a series of notes on an instrument at specified time instants). As such, we showcase much of our work on routing through this medium. Particular to robotic applications, we analyze constraints like maximum velocities that the robots cannot exceed, and information-exchange networks that must remain connected. Furthermore, we consider a notion of heterogeneity where robots and spatial locations are associated with multiple skills, and a robot can visit a location only if it has at least one skill in common with the skill set of that location. To extend the scope of our work, we analyze spatio-temporal routing in the context of a distributed framework, and a dynamic environment.

CHAPTER I

INTRODUCTION

Dissertation Statement: The objective of the research is spatio-temporal routing under various constraints *specific to* multi-robot applications.

Multi-robot routing is a well studied topic in robotics, that requires multiple robots to visit a set of spatially distributed requests for some purpose (e.g., delivery or acquisition) with routes that optimize certain criteria (e.g., minimization of total distance traveled, or energy consumption). However, solving such routing problems, in general, is computationally expensive. Moreover, one can note that many times, applications require that spatial locations be visited in a synchronized and sequenced manner, thus motivating the need for spatio-temporal requests (spatial locations that must be visited *at specified time instants*) in lieu of purely spatial requests. Such a spatio-temporal construction is especially convenient for applying the framework of assignment problems towards finding solutions, with the resulting reduction in complexity: a concept that forms the basis for the work presented in this dissertation.

We consider *music* as an intuitive medium for demonstrating the spatio-temporal theme of our work. For instance, consider a person presented with the task of playing a particular piano piece. The sheet music for such a piece acts as a set of instructions for the person, providing information on which piano notes need playing at which time instants. However, the person needs to figure out "how" to play the piece, for example, decide which finger goes where, or how many fingers should be used. Drawing inspiration from such notions (see Figure 1), we use spatio-temporal requests as a high level instruction set for multiple robots, and analyze the ensuing routing problem under different constraints particular to

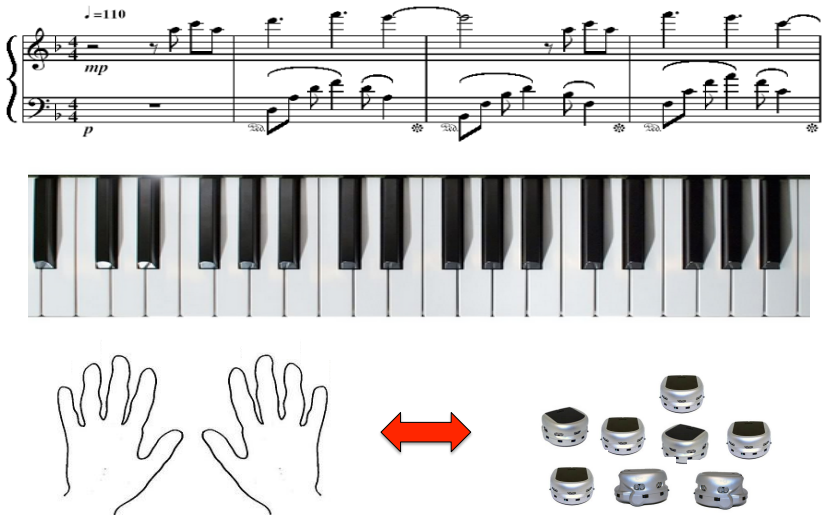


Figure 1: Piano sheet music as high level instructions for a pianist's fingers, or equivalently, a series of spatio-temporal requests as a high level instruction set for multiple robots.

multi-robot applications.

As such, we introduce the *Robot Music Wall* (see Figure 2), a musically instrumented surface where planar positions correspond to distinct notes of an instrument. By perceiving a musical composition as a series of spatio-temporal requests, multiple robots with the ability to traverse the wall can effectively "play" the piece of music by being routed to such requests, demonstrating the spatio-temporal theme central to this dissertation.

In many applications of multi-robot routing like search and rescue, transportation on demand, assembly, and surveillance, several practical constraints are in play, that should be considered in the analysis of problems beforehand. For instance, robots may have constraints on their dynamics, or limited sensing/communication abilities that could affect their ability to accomplish tasks in a collaborative manner, or limited functionalities/skills required for accomplishing different tasks. Furthermore, the choice of applications and infrastructures may dictate the need for employing centralized versus distributed techniques towards finding solutions. Additionally, the task environments in which the robots operate may remain static, or dynamically change over time, further influencing the analysis of

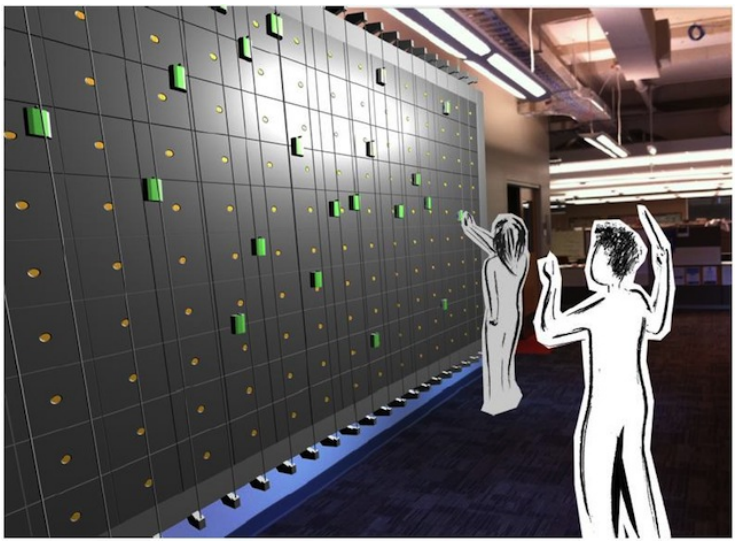


Figure 2: A rendering of the Robot Music Wall concept.

different situations.

Motivated by these facts, our work incorporates constraints like maximum velocity and connectivity maintenance into spatio-temporal routing, where in the first case, robots have a maximum velocity they cannot exceed, while in the second case, robots are range constrained and must ensure a connected information exchange network at all times. Moreover, we inject a notion of heterogeneity in the context of spatio-temporal routing, where each robot, as well as each request, is associated with one or more skills, and a request can be serviced by a robot only if the robot has at least one skill in common with the skill set of that request. In an effort to further generalize spatio-temporal routing in the context of multi-robot applications, we analyze it under a distributed framework, and explore dynamic environments where a user is allowed to modify spatio-temporal requests in real-time.

We demonstrate the entirety of our work through simulations and hardware implementations, centered around the premise of the Robot Music Wall. We conclude this dissertation with a graphical user-interface that allows humans to interact with multiple robots through the creation of different musical compositions.

1.1 Background

Multi-robot routing represents an important class of problems in the robotics literature, with applications in several domains like military, construction, reconnaissance, warehouse automation, transportation on demand, and planetary exploration. In this survey, we review existing methods that address such routing problems. Since we consider routing under a spatio-temporal construction, a considerable body of our work deals with applying the framework of assignment problems towards finding solutions. Thus, we include in this section, a brief review of assignment problems, and in particular, certain formulations that are integrated throughout the fabric of the work presented. Moreover, to provide context for our work concerning the inclusion of different practical constraints, we review connectivity maintenance and heterogeneity in multi-robot routing scenarios. Additionally, towards the end of this dissertation, we analyze assignment problems under a *distributed* framework. Thus, we conclude this section with a review of existing literature on the same.

1.1.1 Multi-Robot Routing

In its simplest form, a multi-robot routing problem is specified by a set of robots $R = \{r_1, r_2, ..., r_n\}$, a set of targets, $T = \{t_1, t_2, ..., t_m\}$, their locations, and a non-negative cost function $c(i, j)$, $i, j \in R \cup T$, which denotes the cost of moving between locations i and j [45]. The objective of multi-robot routing is to find an allocation of targets to robots and a path for each robot that visits all targets allocated to it so that a team objective is optimized.

Many combinatorial optimization problems in the field of operations research can be extended towards finding centralized solutions to multi-robot routing problems. For instance, Bektas [4] provides an extensive review of the Multiple Traveling Salesman Problem (m-TSP), that consists of determining a set of optimal routes for m salesmen who all start from and turn back to a home city. Furthermore, he details variations of the m-TSP concerning multiple depots, variable number of salesmen, and time windows, that can be

associated with many real world routing problems arising in the planning of autonomous mobile robots [8, 72, 77], and unmanned aerial vehicles [62]. Another problem closely related to multi-robot routing is the well known Vehicle Routing Problem (VRP) [7, 74], which concerns the design of optimal delivery or collection routes for a fleet of vehicles from one or many depots to a number of geographically scattered customers with known demands. The dynamic counterpart of the VRP deals with online arrival of customer demands during the operation [9, 54], while variations like capacitated VRP [60] and VRP with time windows [70] incorporate practical constraints like capacities and timely deliveries.

In addition to centralized methods, many decentralized approaches that use market-based mechanisms, in particular, auction-based methods [22, 75], have been developed for addressing multi robot routing problems. In such methods, the communicated information consists of bids robots place on various tasks, and coordination is achieved by a process similar to winner determination in auctions.

Solving such routing problems in general is computationally expensive. In fact, Lagoudakis et al. [45] show that the multi-robot routing problem under team objectives like the sum of robot path costs over all robots (total distance traveled), or the maximum robot path cost over all robots, is NP-hard. Similarly, the m-TSP as well as the the VRP are both proven to be NP-hard [34, 40]. This drawback regarding the complexity of multi-robot routing is mitigated in our work in Chapters 2 and 3.1, through the introduction of temporal constraints.

1.1.2 Assignment Problems

Assignment problems are an integral part of combinatorial optimization, with wide applicability in theory as well as practice [43, 45, 53]. As discussed previously, assignment problems play a fundamental role throughout the analysis of our work. In general, the term "assignment problem" (AP) is recognized to have originated from Kuhn's article [44]. He

describes the problem as follows: for a given set of "tasks" and a given set of "agents", the objective is to ensure each task is assigned to a different agent, with each agent being assigned at most one task (a one-to-one assignment), while minimizing the total cost of the assignment. Over the past few decades, many variations to the assignment problem have emerged, including problems with different or multiple objectives [27, 33], and problems that involve one-to-many [2] or many-to-one matching [76]. In an extensive survey, Pentico [55] enumerates such variations, in conjunction with their applications and methods used for solving them.

Model with one task per agent: In this section, we review the classic AP, also called the *Linear Sum Assignment Problem* (LSAP). The mathematical model for the classic AP is given as:

$$\min_{x} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j} \tag{1}$$

subject to:

$$x_{i,j} \in \{0, 1\} \tag{2}$$

$$\sum_{i=1}^{n} x_{i,j} = 1, \, \forall \, j \in \{1, ..., n\} \tag{3}$$

$$\sum_{j=1}^{n} x_{i,j} = 1, \, \forall \, iin\{1, ..., n\} \tag{4}$$

where $x_{i,j} = 1$ if agent i is assigned to task j, 0 if not, and $c_{i,j}$ is the cost of assigning agent i to task j. Martello [49] provides a survey that enumerates the many algorithms, both sequential and parallel, that have been developed for solving the LSAP, ranging from primal-dual combinatorial algorithms to simplex-like methods, cost operation algorithms, forest algorithms, and relaxation approaches. In our work in Chapters 2, 3.1, 3.2 and 4, we model different problems and sub-problems as variants of the LSAP, and choose to use the first polynomial-time primal-dual algorithm developed for solving the LSAP, called the

Hungarian Method [44]. Note that the fastest version of the *Hungarian Method* involving N stages is $\mathcal{O}(N^3)$ (see implementation in [46]).

Model with multiple tasks per agent: Another popular variation of the AP is the *Generalized Assignment Problem* (GAP) [68] that allows an agent to be assigned to multiple tasks. A large body of work exists on analyzing the GAP, and its variants involving time minimization [2], bottleneck constraints [50], and cardinality constraints [13]. The mathematical model for the GAP is given as:

$$\min_{x} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j} \tag{5}$$

subject to:

$$x_{i,j} \in \{0, 1\} \tag{6}$$

$$\sum_{i=1}^{n} x_{i,j} = 1, \, \forall \, j \in \{1, ..., n\} \tag{7}$$

$$\sum_{j=1}^{n} a_{i,j} x_{i,j} \leq b_{i,j}, \, \forall \, iin\{1, ..., n\} \tag{8}$$

where $x_{i,j} = 1$ if agent i is assigned to task j, 0 if not, and $c_{i,j}$ is the cost of assigning agent i to task j, $a_{j,j}$ is the amount of agent $i's$ capacity used if that agent is assigned to tai j, and b_i is the available capacity of agent i. Our work concerning heterogeneity in Chapter 4 exploits the above mentioned model with multiple tasks per agent.

1.1.3 Connectivity Maintenance

An important aspect of multi-robot coordination concerns connectivity maintenance, where to ensure that the robots can execute a mission in a collaborative manner, the induced information exchange network must be sufficiently rich. Note that the information network may pertain to either sensing or communication capabilities, or both. Though connectivity constraints add a new level of complexity to the task allocation problem, they bring the

multi-robot routing problem closer to reality. For instance, most prior work on multi-robot routing, implicitly or explicitly, makes the assumption that the robots of the team are able to communicate at all times independently of their present location, a fact that cannot be taken for granted in many interesting real-world applications.

In general, connectivity maintenance in multi-robot networks requires techniques for ensuring connectivity of a range constrained multi-robot network during some task execution. Maintaining full connectivity between team members does not necessarily imply that each robot is connected directly with all other robots, but rather that the minimum spanning tree over all robot locations has no edge longer than the maximum connectivity radius [32].

A popular technique in the robotics literature, that addresses the issue of connectivity, is the deployment of "relay" robots, that form chain structures to maintain connectivity [24, 31, 51]. Other methods involve modeling the connectivity constraints as formation constraints, and employing potential field controllers based on navigation functions, towards motion planing [39, 56]. Basic approaches opportunistically take advantage of network connectivity when available [10], but do not explicitly avoid network splits, while other methods seek connectivity at *particular* time instants only [58, 59]. We address the issue of connectivity under a spatio-temporal construction in Section 3.2, where we construct routes that maintain connectivity *for all times*, while allowing dynamic assignment between robots and spatio-temporal requests such that no robots exist *solely* for the task of maintaining connectivity links.

1.1.4 Heterogeneity

A number of multi-robot applications require a mixture of robotic capabilities, that is too extensive to design in a single robot. Moreover, time constraints may require the use of multiple robots working simultaneously on different aspects of the mission (routing for instance), to successfully accomplish the objective. As such, "heterogeneity" in robots is

a coveted feature, that can be utilized towards task completion. However, heterogeneity can be characterized through several different metrics. For instance, in [26], robots are heterogeneous in their ability to traverse the environment in that they differ on the basis of movement speed and task execution speed, while in [37], heterogeneity is encoded through different sensing and manipulation techniques. In [57], robots are classified into groups based on varying sizes of sensor footprints.

A higher layer of classification seen in many heterogeneous robotic applications, is provided through different multi-robot task allocation models as per the taxonomy in [29]. For instance, in [65, 66] , the model is such that each robot can perform at most a single task at a given time, and each task requires to be completed by a single robot, while in [41, 64], the model requires that each task be completed by multiple robots. Methods for approaching such heterogeneous problem include algorithms based on gossip [26], token passing [65] and Mixed Integer Programming [41, 61]. Though these methods provide quality guarantees, they do not consider explicit temporal constraints on the spatial tasks requiring servicing. In Chapter 4, we explore heterogeneity under such a spatio-temporal construction. In particular, we characterize heterogeneity in terms of physical abilities or skills possessed by the robots, and consider the model where each robot can service at most one request at a given time, and each request can be serviced by exactly one robot.

1.1.5 Distributed Assignment

As discussed before, assignment problems are very popular in the field of distributed control, and often comprise of finding a one-to-one matching between multiple robots and tasks, while minimizing some assignment benefit, for instance, the total distance traveled [38, 69]. The cost for global computation and information in such scenarios can be prohibitively high, creating the need for local solutions, a task often challenging due to limited communication capabilities, and global knowledge of the robots.

Various techniques have been proposed for solving the multi-robot assignment problem (or multi-robot task allocation (MRTA) problem) in both centralized and distributed settings. Among centralized algorithms [3, 5, 21], the Hungarian Method [44] was the first to compute an optimal solution in finite time. In [30], the authors propose a distributed version of the Hungarian Method. However, they require a coordinator (root) robot, as well as a predetermined communication network, on each iteration of their algorithm. Moreover, no convergence guarantees are provided in [30]. Other decentralized techniques involve consensus based approaches that typically require that robots converge on a consistent situational awareness before performing the assignment [1, 23, 73]. Though such methods are robust, they are typically slow to converge, and require the transmission of large amounts of data.

Distributed methods that solve linear programs, for instance, can also be employed towards solving assignment problems [11, 25], though they are computationally expensive, especially in comparison to more streamlined algorithms, developed for the purpose of solving assignment problems. Other popular techniques include auction algorithms that require robots to bid on tasks, rendering them more or less attractive based on the corresponding prices computed [6, 28]. Auction algorithms, though computationally efficient, usually require a coordinator or shared memory. In [79], the authors develop an auction algorithm without such constraints, however with high worst-case convergence bounds. In Section 5.1, we develop our own distributed version of the Hungarian Method, using a connectivity assumption, where robots communicate locally with adjacent robots via a dynamic connected directed information exchange network, and all robots converge to identical optimal solutions, without any coordinator or shared memory.

We extend the distributed assignment algorithm to its dynamic counterpart, and incorporate it into spatio-temporal routing. In particular, we generate online routes for the robots by *iteratively* assigning the robots to positions specified at successive time instants.

In essence, such a routing scheme is similar to online multi-robot task allocation (OM-RTA) [47, 63, 78]. We differ from previous techniques on the basis of the above mentioned algorithm that we develop for distributed task allocation.

1.2 Contributions

Through this dissertation, we analyze spatio-temporal routing under various constraints specific to the multi-robot domain. In Chapter 2, we introduce the basic problem of spatio-temporal routing, and develop an approach embedded in the theory of assignment problems, towards solving it. In Chapter 3, we incorporate practical constraints like capped robot velocities and network connectivity into the above mentioned basic problem, while in Chapter 4, we enable robots with heterogeneous capabilities. In each of the above mentioned cases, we provide results on feasibility and the minimum number of robots required, as well as generate explicit trajectories for the robots to execute. In Chapter 5, we provide an algorithm for solving the assignment problem under a distributed framework, with precise results on convergence, complexity and message size. Additionally, we extend the algorithm towards *dynamic* spatio-temporal routing, enabling an autonomous team of robots to determine their routes online, in a distributed manner, while the spatio-temporal requests are modified (by a user) in real-time. For each chapter mentioned above, we demonstrate our results musically, through the concept of the Robot Music Wall. Finally, we provide some concluding remarks in Chapter 6.

CHAPTER II

SPATIO-TEMPORAL ROUTING - THE BASIC PROBLEM

In this chapter, we consider the basic spatio-temporal routing problem, where multiple robots are required to visit a series of spatial locations at specified time instants (spatio-temporal requests). In keeping with the musical parallel drawn previously in the Introduction, we call such a series of spatio-temporal requests, a *Score*. For convenience, we refer to spatio-temporal requests as *timed positions*.

We let $T = \{t_1, t_2, ..., t_n\}$ denote the set of n discrete time instants over which the Score is defined, where $t_1 < ... < t_n$. Moreover, we let P_i denote the corresponding set of planar positions that require simultaneous servicing at time t_i. Each position in this set is denoted by $P_{i,\alpha}$, where $\alpha \in \{1, ..., |P_i|\}$ (the symbol $|\cdot|$ denotes cardinality), i.e.,

$$P_i = \{P_{i,\alpha} \,|\, \alpha \in \{1, ..., |P_i|\}\}, \quad \forall i \in \{1, ..., n\} \tag{9}$$

We let $\mathcal{K}$ be the maximum number of timed positions that require simultaneous servicing at any time instant in T, given by $\mathcal{K} = \max_{i \in \{1,...,n\}} |P_i|$.

Definition 1 *Let the **Score**, denoted by Sc, be the set of all timed positions that the robots must reach. We express such timed positions as (position, time) pairs, i.e.,*

$$Sc = \{(P_{i,\alpha}, t_i) \,|\, i \in \{1, ..., n\}, \alpha \in \{1, ..., |P_i|\}\} \tag{10}$$

Moreover, for a given set of r robots, denoted by $R = \{1, ..., r\}$, we let $P_0 = \{P_{0,\alpha} \,|\, \alpha \in \{1, ..., |P_0|\}\}$ be the set of their initial positions, defined at time instant t_0.

For a given (Sc, R, P_0), we are interested in the problem of routing the robots to reach the timed positions contained in the Score, with the total distance traveled as the optimization criterion. We refer to this problem as the **Unconstrained Routing Problem (URP)**

[16, 19]. Note that we want our solution to act at a high enough level of abstraction so that the dynamics of the robots do not have to be explicitly accounted for. Thus, for convenience, we represent each robot $p \in R$ as a planar point particle with single integrator dynamics, given by $\dot{x}_p = u_p$. For such systems, since minimum distance paths are straight lines and minimum energy motions have constant velocities, we let robots move between assigned positions in straight line paths with constant velocities that ensure their timely arrival.

2.1 What Constitutes Feasible Routes

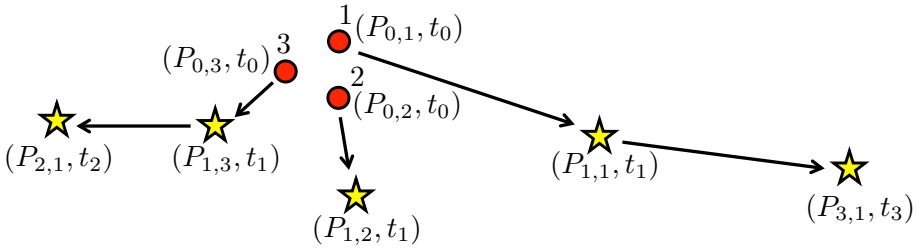
Since we can interpret the path of any robot as a *series of individual assignments* between timed positions assigned to that robot, directed in increasing order of specified time instants, the information contained in the paths of the robots can be encoded in a function that explicitly describes such individual assignments. We address the feasibility aspects of the URP through such a notion of individual assignments.

Definition 2 *Let the **Assignees**, denoted by As, be the set containing all timed positions in the Score specified before the last time instant t_n, in addition to all timed initial positions of the robots, i.e.,*

$$As = \{(P_{i,\alpha}, t_i) \,|\, i \in \{0, ..., n-1\}, \alpha \in \{1, ..., |P_i|\}\} \tag{11}$$

For convenience, let $\mathcal{I} \triangleq \{0, ..., n-1\}$ and $\mathcal{J} \triangleq \{1, ..., n\}$ be the index sets for i and j, representing the time instants at which positions are specified in As and Sc respectively. Also, let $\mathcal{A}_i \triangleq \{1, ..., |P_i|\}$, $i \in \{0, ..., n\}$ be the index set for α and β.

Definition 3 *The triple (Sc, R, P_0) is **feasible** if there exists some $As' \subseteq As$, such that the function $\pi : As' \to Sc$ is a bijection, where $\pi((P_{i,\alpha}, t_i)) = (P_{j,\beta}, t_j) \in Sc \Rightarrow t_j > t_i$ for all $(P_{i,\alpha}, t_i) \in As'$.*



$Sc = \{(P_{1,1}, t_1), (P_{1,2}, t_1), (P_{1,3}, t_1), (P_{2,1}, t_2), (P_{3,1}, t_3)\}$
$R = \{1, 2, 3\}$
$P_0 = \{P_{0,1}, P_{0,2}, P_{0,3}\}$
$As = \{(P_{0,1}, t_0), (P_{0,2}, t_0), (P_{0,3}, t_0), (P_{1,1}, t_1), (P_{1,2}, t_1), (P_{1,3}, t_1), (P_{2,1}, t_2)\}$
$As' = \{(P_{0,1}, t_0), (P_{0,2}, t_0), (P_{0,3}, t_0), (P_{1,1}, t_1), (P_{1,3}, t_1)\}$

$\pi : As' \rightarrow Sc$
$\pi((P_{0,1}, t_0)) = (P_{1,1}, t_1)$
$\pi((P_{1,1}, t_1)) = (P_{3,1}, t_3)$
$\pi((P_{0,2}, t_0)) = (P_{1,2}, t_1)$
$\pi((P_{0,3}, t_0)) = (P_{1,3}, t_1)$
$\pi((P_{1,3}, t_1)) = (P_{2,1}, t_2)$

Figure 3: An example of a function π as per Definition 3, for a given feasible triple (Sc, R, P_0) (robots are depicted as circles, and timed positions in the Score as stars.)

Notice that the function π maps between timed positions in the Assignees and the Score, and ensures that every timed position in the Score is assigned, no two timed positions in the Assignees are mapped to the same timed position in the Score, and no two timed positions in the Score are assigned to the same timed position in the Assignees. Moreover, it enforces directionality within each individual assignment, i.e. it ensures that each timed position in the Score is assigned to a timed position in the Assignees, specified at some earlier time instant. We call this the directionality constraint. Figure 3 shows an example of such a function π. Notice that for each robot, we can construct its corresponding path by applying π iteratively, starting with the initial position of that robot. However, to ensure the existence of such a π (i.e. establish feasibility), we must have a minimum number of robots. We provide this result in the form of the following Lemma:

Lemma 1 *Given the Score, the minimum number of robots required, $r^\star$, to ensure that the corresponding triple $(Sc, R^\star, P_0)$ is feasible, where $R^\star = \{1, ..., r^\star\}$ is the set of robots , and P_0 is the set of their initial positions, is given by $\mathcal{K}$ (where $\mathcal{K}$ is the maximum number of positions that require simultaneous servicing in the Score).*

Proof If $r < \mathcal{K}$, i.e. we have fewer robots than the maximum number of timed positions requiring simultaneous servicing in the Score, then all $\mathcal{K}$ positions cannot be reached simultaneously. Consequently, a function π as per Definition 3 cannot exist. For the case $r \geq \mathcal{K}$, though intuitively trivial, we proceed to prove the existence of such a π. Note that $r \geq \mathcal{K}$ implies that $|As| \geq |Sc|$. Without loss of generality, assume $r = \mathcal{K}$, and $|As| = |Sc|$.

We generate a $0-1$ element cost matrix $C = [c(i, \alpha, j, \beta)]$ of size $|As| \times |Sc|$, where the rows and columns in C represent timed positions in the Assignees and the Score respectively, and the element $c(i, \alpha, j, \beta) = 1 \iff t_j > t_i$, and 0 otherwise. Note that we can denote the element $c(i, \alpha, j, \beta)$ by $c(p, q)$, where $p, q \in \{1, ..., |As|\}$, and,

$$p = \sum_{\theta=0, i>0}^{i-1} |P_\theta| + \alpha \tag{12}$$

$$q = \sum_{\theta=1, j>1}^{j-1} |P_\theta| + \beta \tag{13}$$

Now consider a bijective function $\pi : As \rightarrow Sc$, such that $\pi((P_{i,\alpha}, t_i)) = (P_{j,\beta}, t_j) \Rightarrow c(i, \alpha, j, \beta) = c(p, q), (p = q)$, i.e. π contains individual assignments between all timed positions in As and Sc that have a corresponding diagonal cost entry in the cost matrix C. In the remainder of this proof, we will show (through contradiction) that such a function π satisfies the condition for feasibility as per Definition 3, or in other words, π does not contain any individual assignment that violates the directionality constraint.

Let us assume there exists an individual assignment in π that violates the directionality constraint, i.e. there exists a diagonal element $c(p, p) = c(i, \alpha, j, \beta)$ in C with corresponding cost 0. By construction, this implies that $j \leq i$. Immediately, we notice that for $i = 0$, there exists no $j \in \mathcal{J}$ such that $j \leq i$. Similarly, for $j = n$, there exists no $i \in \mathcal{I}$ such

that $j \leq i$. As a result, we see that $c(p,p) = c(i,\alpha,j,\beta) = 0$ is a contradiction for the above values of i and j. Hence, we direct our focus on the remaining values of i and j, i.e. $i \in \{1,...,n-1\}$ and $j \in \{1,...,n-1\}$.

Notice that we can rewrite (12) for index p as follows,

$$p = |P_0| + \sum_{\theta=1,j>1}^{j-1} |P_\theta| + \sum_{\theta=j,i>j}^{i-1} |P_\theta| + \alpha \tag{14}$$

$$p = r + \sum_{\theta=1,j>1}^{j-1} |P_\theta| + \sum_{\theta=j,i>j}^{i-1} |P_\theta| + \alpha \tag{15}$$

Since $p = q$, we can equate (15) and (13) to get the following,

$$r + \sum_{\theta=j,i>j}^{i-1} |P_\theta| + \alpha = \beta \tag{16}$$

Moreover, $r = K \Rightarrow r \geq \beta$, since $1 \leq \beta \leq |P_j| \leq \mathcal{K}, \quad \forall j \in \{1,...,n-1\}$, thereby making (16) a contradiction. Hence, we have shown that there exists no diagonal element in C with corresponding cost 0, or that π is a bijective function as per Definition 3. ∎

2.2 Finding Optimal Routes

Once we establish feasibility, in that we ensure we have enough robots ($r \geq r^\star$), we can focus on the problem of finding the paths of the robots. As mentioned previously, we associate with such paths, the total distance traveled as a cost to be minimized. Thus, for a given (Sc, R, P_0) that is feasible, we would like to find a function π as per Definition 3, with minimum total distance traveled. We call such a function an *optimal assignment*, and denote it by $\pi^\star$.

By defining a mapping $l(i,\alpha,j,\beta)$, we can formulate the problem of finding such an optimal assignment $\pi^\star$, as a linear program described as follows:

$$\min_{l} \sum_{i \in \mathcal{I}} \sum_{\alpha \in \mathcal{A}_i} \sum_{j=i+1}^{n} \sum_{\beta \in \mathcal{A}_j} ||P_{j,\beta} - P_{i,\alpha}||\, l(i,\alpha,j,\beta) \tag{17}$$

subject to:

$$l(i, \alpha, j, \beta) \in \{0, 1\} \tag{18}$$

$$\sum_{i=0}^{j-1} \sum_{\alpha \in \mathcal{A}_i} l(i, \alpha, j, \beta) = 1, \forall\, j \in \mathcal{J}, \beta \in \mathcal{A}_j \tag{19}$$

$$\sum_{j=i+1}^{n} \sum_{\beta \in \mathcal{A}_j} l(i, \alpha, j, \beta) \leq 1, \forall\, i \in \mathcal{I}, \alpha \in \mathcal{A}_i \tag{20}$$

where $l(i, \alpha, j, \beta)$ represents the individual assignment of $(P_{i,\alpha}, t_i) \in As$ to $(P_{j,\beta}, t_j) \in Sc$, and is 1 if the assignment is done, and 0 otherwise. The resulting l gives us the corresponding optimal assignment $\pi^\star$, where $l(i, \alpha, j, \beta) = 1 \iff \pi^\star((P_{i,\alpha}, t_i)) = (P_{j,\beta}, t_j)$.

The linear program in Equations (17) - (20) represents the URP, and is a modified version of the classic *Linear Sum Assignment Problem* (LSAP) [49]. In particular, the LSAP concerns the following: given two equal sized sets P and Q with some non-negative cost function $\mathcal{C} : (P \times Q) \rightarrow \mathbb{R}$, the objective is to find a bijection $S : P \rightarrow Q$ that minimizes the function $\sum_{a \in P} \mathcal{C}(a, S(a))$. As and Sc in the URP correspond to P and Q in the LSAP, and the function π as per Definition 3 corresponds to S. Note that the LSAP insists on P and Q being equal sized while the URP insists on $|As| \geq |Sc|$. Moreover, in the LSAP, there exist no forbidden individual assignments between P and Q, contrary to the URP, where individual assignments between As and Sc that violate the directionality constraint are forbidden. However, we can apply algorithms developed for solving the LSAP towards solving the URP, by incorporating certain modifications that we discuss later in this section.

Many algorithms, both sequential and parallel, have been developed for solving the LSAP (e.g. [49]), ranging from primal-dual combinatorial algorithms to simplex-like methods, cost operation algorithms, forest algorithms, and relaxation approaches. Although immaterial to the underlying theory, in this paper, we choose to use the first polynomial-time primal-dual algorithm developed for solving the LSAP, called the *Hungarian Method* (see [44]). Note that the fastest version of the Hungarian Method involving N stages is $\mathcal{O}(N^3)$

(see implementation in [46]).

To use the Hungarian Method towards solving the URP, we generate a cost matrix $C = [c(i, \alpha, j, \beta)]$ of size $|As| \times |Sc|$, where the rows and columns in C represent timed positions in the Assignees and the Score respectively, and the element $c(i, \alpha, j, \beta)$ equals the distance between the corresponding timed positions, i.e. $||P_{j,\beta} - P_{i,\alpha}||$, $(P_{i,\alpha}, t_i) \in As$ and $(P_{j,\beta}, t_j) \in Sc$. The Hungarian Method requires a square cost matrix, for which we introduce $(|As| - |Sc|)$ *dummy*[1] positions as targets (in addition to the timed positions in the Score). For convenience, we denote the set of such *dummy* positions by $P_{n+1} = \{P_{n+1,\beta} \,|\, \beta \in \mathcal{A}_{n+1}\}$ where $\mathcal{A}_{n+1} \triangleq \{1, 2, ..., (|As| - |Sc|)\}$. Moreover, we let the cost associated with reaching these *dummy* positions be zero. We define Sc' to be the set containing the Score in addition to the *dummy* positions, i.e., $Sc' = Sc \cup \{(P_{n+1,\beta}, t_{n+1}) \,|\, \beta \in \mathcal{A}_{n+1}\}$. Forbidden individual assignments are typically dealt with, by associating a prohibitively large cost $\mathfrak{M}$ with each of them[2]. We denote this modified cost matrix by $\hat{C} = [\hat{c}(i, \alpha, j, \beta)]$, where,

$$\hat{c}(i, \alpha, j, \beta) = \begin{cases} ||P_{j,\beta} - P_{i,\alpha}||, & i \in \mathcal{I}, \alpha \in \mathcal{A}_i, \\ & j \in \{i+1, ..., n\}, \beta \in \mathcal{A}_j \\ \mathfrak{M}, & i \in \mathcal{I}, \alpha \in \mathcal{A}_i, \\ & j \in \{1, ..., i\}, \beta \in \mathcal{A}_j \\ 0, & i \in \mathcal{I}, \alpha \in \mathcal{A}_i, \\ & j = n+1, \beta \in \mathcal{A}_j \end{cases} \tag{21}$$

The symbol $\mathfrak{M}$ represents forbidden individual assignments. The Hungarian Method operates on such a square cost matrix to search for a one-to-one correspondence (bijective function) between its row and column elements, with a minimum total cost. Since the triple (Sc, R, P_0) is feasible, we know there exists a bijective function as per Definition 3 (e.g.

[1]The term *dummy* is standard in the assignment literature, e.g. see [49].

[2]As per the big-M method [12, 55])

the function π as shown in the proof of Lemma 1). Consequently, the Hungarian Method always finds a bijective function $H_r : As \rightarrow Sc'$ with a minimum total cost, *which avoids forbidden individual assignments*. Thus, using H_r, we can construct an optimal assignment $H_r|_{As'} : As' \rightarrow Sc$, denoted by $H_r^\star$, where $As' \subseteq As$ is the set of timed positions in the Assignees that are not assigned to *dummy* positions. By restricting the total number of robots to be no greater than the total number of timed positions in the Score, i.e., $r \leq |Sc|$, the Hungarian Method finds the optimal paths for the robots, with complexity $\mathcal{O}(|Sc|^3)$.

CHAPTER III

CONSTRAINED SPATIO-TEMPORAL ROUTING

We extend our analysis of the Unconstrained Routing Problem (URP), to its constrained counterparts, where we consider the following two practical scenarios: first, the robots have a maximum velocity that they cannot exceed (Section 3.1), and second, the robots are range constrained and must maintain a connected information exchange network at all times (Section 3.2).

3.1 Capping Robot Velocities

So far, we assume that robots do not have any constraints on their velocities. However, a more realistic approach would be to introduce, say, a maximum velocity $\hat{v}$ that the robots cannot exceed while driving between assigned timed positions. We refer to the resulting problem of routing velocity constrained robots to reach the timed positions contained in the Score, while optimizing the total distance traveled, as the **Velocity Constrained Routing Problem (VCRP)** [16]. Similar to the Unconstrained Routing Problem (URP), we approach feasibility of the VCRP in terms of a function $\hat{\pi}$ that describes individual assignments between timed positions in the Score.

Definition 4 *The quadruple* $(Sc, R, P_0, \hat{v})$ *is* ***feasible*** *if there exists some* $As' \subseteq As$*, such that the function* $\hat{\pi} : As' \rightarrow Sc$ *is a bijection, where* $\hat{\pi}((P_{i,\alpha}, t_i)) = (P_{j,\beta}, t_j) \in Sc \Rightarrow t_j > t_i \wedge \frac{||P_{j,\beta} - P_{i,\alpha}||}{t_j - t_i} < \hat{v}$ *for all* $(P_{i,\alpha}, t_i) \in As'$.

Note that for a given Score and a maximum velocity $\hat{v}$, the minimum number of robots required, $r^\star$, that ensures that the resulting quadruple $(Sc, R^\star, P_0, \hat{v})$ is feasible, where

$R^\star = \{1, ..., r^\star\}$ is the set of robots , and P_0 is the set of their initial positions, depends on the initial positions of the robots. Thus, to remove this dependence, we assume the following,

Assumption 1 *The starting position of every robot is chosen such that it can reach any timed position in the Score without violating the maximum velocity constraint.*

Moreover, from Lemma 1, we know that irrespective of the maximum velocity $\hat{v}$, we require *at least* $\mathcal{K}$ robots, i.e. $r^\star \geq \mathcal{K}$ (where . Thus, for a given $(Sc, R, P_0, \hat{v})$, where $R = \{1, 2, ..., \mathcal{K}\}$, we can pose the problem of finding $r^\star$ as follows:

By defining a $0-1$ element cost matrix $C = [c(i, \alpha, j, \beta)]$ of size $|As| \times |Sc|$, where $c(i, \alpha, j, \beta) = 1 \iff (t_j > t_i) \wedge (\frac{||P_{j,\beta} - P_{i,\alpha}||}{t_j - t_i} < \hat{v})$, we state the following linear program,

$$\max_{l} \sum_{i \in \mathcal{I}} \sum_{\alpha \in \mathcal{A}_i} \sum_{j=i+1}^{n} \sum_{\beta \in \mathcal{A}_j} c(i, \alpha, j, \beta) l(i, \alpha, j, \beta) \tag{22}$$

subject to:

$$l(i, \alpha, j, \beta) \in \{0, 1\} \tag{23}$$

$$\sum_{i \in \mathcal{I}} \sum_{\alpha \in \mathcal{A}_i} l(i, \alpha, j, \beta) = 1, \, \forall \, j \in \mathcal{J}, \beta \in \mathcal{A}_j \tag{24}$$

$$\sum_{j \in \mathcal{J}} \sum_{\beta \in \mathcal{A}_j} l(i, \alpha, j, \beta) \leq 1, \, \forall \, i \in \mathcal{I}, \alpha \in \mathcal{A}_i \tag{25}$$

Note that the linear program shown above is another modified version of the Linear Sum Assignment Problem (LSAP), and can be solved using many techniques mentioned earlier in Section 2.2, including the Hungarian Method. Consequently, the minimum number of robots, $r^\star$ is given by,

$$r^\star = \mathcal{K} + |\{(P_{j,\beta}, t_j) \,|\, c(i, \alpha, j, \beta) = 0 \wedge l(i, \alpha, j, \beta) = 1\}| \tag{26}$$

Once feasibility is established, we can apply methods similar to the ones employed for solving the URP, towards finding routes for the VCRP. In particular, for a given feasible

quadruple $(Sc, R, P_0, \hat{v})$, we can find an optimal assignment $\hat{\pi}^\star$ (a function $\hat{\pi}$ as per Definition 4, with minimum total distance traveled) by solving the following linear program,

$$\min_{L} \sum_{i \in \mathcal{I}} \sum_{\alpha \in \mathcal{A}_i} \sum_{j=i+1}^{n} \sum_{\beta \in \mathcal{A}_j} ||P_{j,\beta} - P_{i,\alpha}||\, l(i, \alpha, j, \beta) \tag{27}$$

subject to:

$$l(i, \alpha, j, \beta) \in \{0, 1\} \tag{28}$$

$$\sum_{i=0}^{j-1} \sum_{\alpha \in \mathcal{A}_i} l(i, \alpha, j, \beta) = 1, \, \forall \, j \in \mathcal{J}, \beta \in \mathcal{A}_j \tag{29}$$

$$\sum_{j=i+1}^{n} \sum_{\beta \in \mathcal{A}_j} l(i, \alpha, j, \beta) \leq 1, \, \forall \, i \in \mathcal{I}, \alpha \in \mathcal{A}_i \tag{30}$$

$$\frac{||P_{j,\beta} - P_{i,\alpha}||}{t_j - t_i} < \hat{v} \iff l(i, \alpha, j, \beta) = 1 \tag{31}$$

Using a similar approach to before, we solve the VCRP by applying the Hungarian Method on an associated cost matrix $\hat{CV} = [\hat{cv}(i, \alpha, j, \beta)]$, where,

$$\hat{cv}(i, \alpha, j, \beta) = \begin{cases} ||P_{j,\beta} - P_{i,\alpha}||, & \frac{||P_{j,\beta} - P_{i,\alpha}||}{t_j - t_i} \leq \hat{v}, \\ & i \in \mathcal{I}, \alpha \in \mathcal{A}_i \\ & j \in \{i+1, ..., n\}, \beta \in \mathcal{A}_j \\ \mathfrak{M}, & i \in \mathcal{I}, \alpha \in \mathcal{A}_i \\ & j \in \{1, ..., i\}, \beta \in \mathcal{A}_j \\ 0, & i \in \mathcal{I}, \alpha \in \mathcal{A}_i \\ & j = n+1, \beta \in \mathcal{A}_j \\ \mathfrak{M}, & \frac{||P_{j,\beta} - P_{i,\alpha}||}{t_j - t_i} > \hat{v}, \\ & i \in \mathcal{I}, \alpha \in \mathcal{A}_i \\ & j \in \{i+1, ..., n\}, \beta \in \mathcal{A}_j \end{cases} \tag{32}$$

Note that we include individual assignments that not only violate the directionality constraint, but also the maximum velocity constraint, in the set of forbidden individual assignments. As before, we denote such assignments by $\mathfrak{M}$ in the cost matrix.

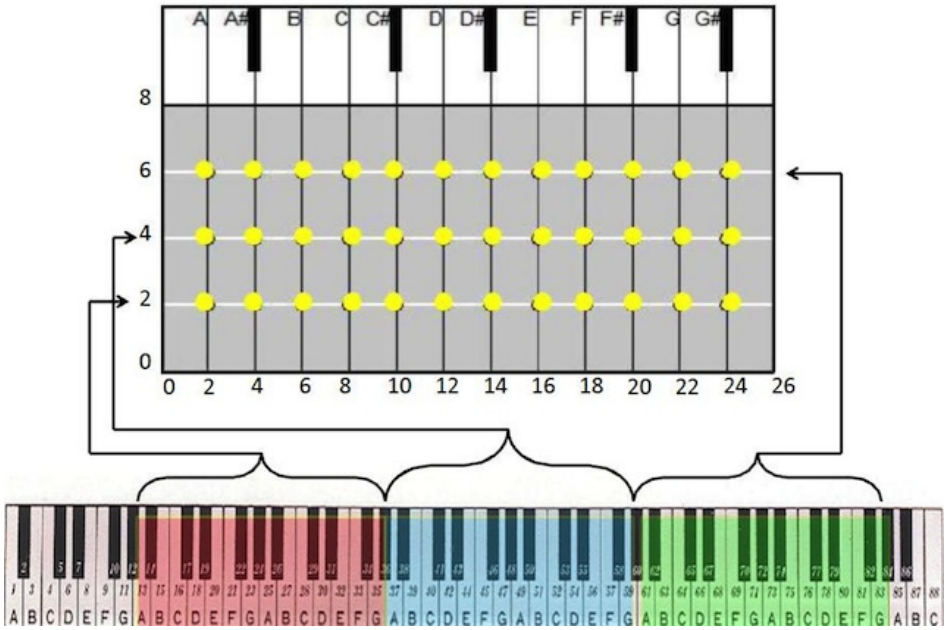
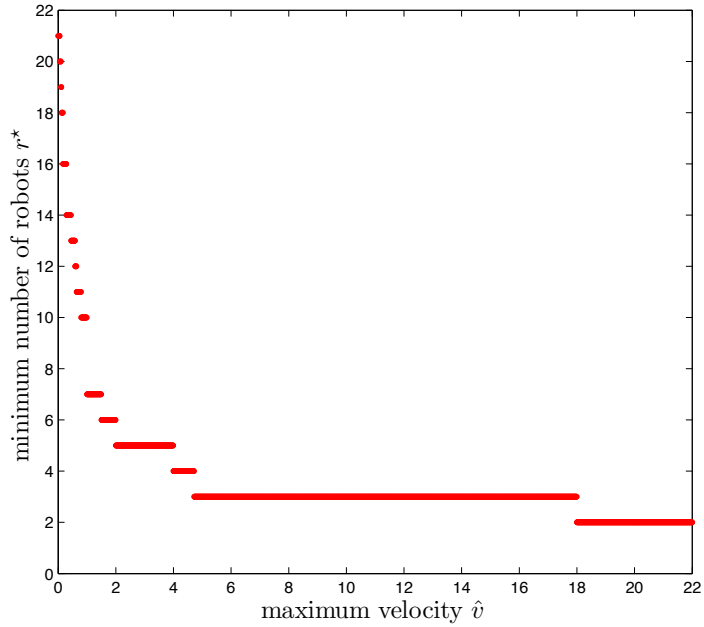


Figure 4: The Robot Music Wall 1.0 (Piano Wall) with 36 coordinates (yellow circles) representing the notes across three octaves of a piano.

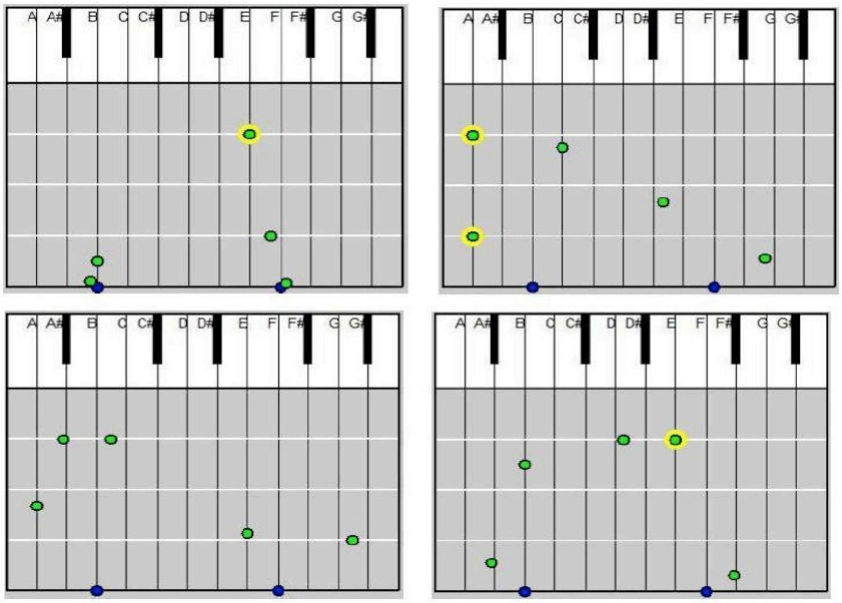
3.1.1 Simulation and Hardware Results (Robot Music Wall 1.0)

In this section, we present the first simulated version of the Robot Music Wall, developed in MATLAB and instrumented to sound like a piano (see Figure 4). We call this wall the *Piano Wall*. Our goal was to make multiple robots perform the popular composition "Für Elise" by Ludwig van Beethoven on this *Piano Wall*. Note that firstly, all notes in the version of "Für Elise" we presented to the robots, lay amongst the set of notes used to create the *Piano Wall*. Secondly, a pianist was required to hit a maximum of two keys simultaneously throughout the performance of the piece ($\mathcal{K} = 2$). With this set-up, we created the Score associated with "Für Elise", containing timed positions on the wall corresponding to notes in "Für Elise", specified at a beat of one second (the smallest time interval between consecutive timed positions).

As depicted in Figure 5b, we simulated the robots as circles (green) on the Piano Wall. Additionally, we created 2 base stations at the bottom of the wall (shown as two circles



(a) A plot of the minimum number of robots $(r^\star)$ versus the maximum velocity $(\hat{v})$.



(b) Four snapshots of the velocity constrained case: $\hat{v} = 3$, $r^\star = 5$, $r = 5$.

Figure 5: Simulation results for Velocity Constrained Routing.

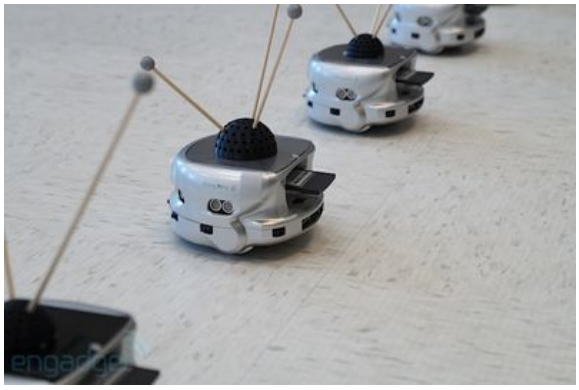


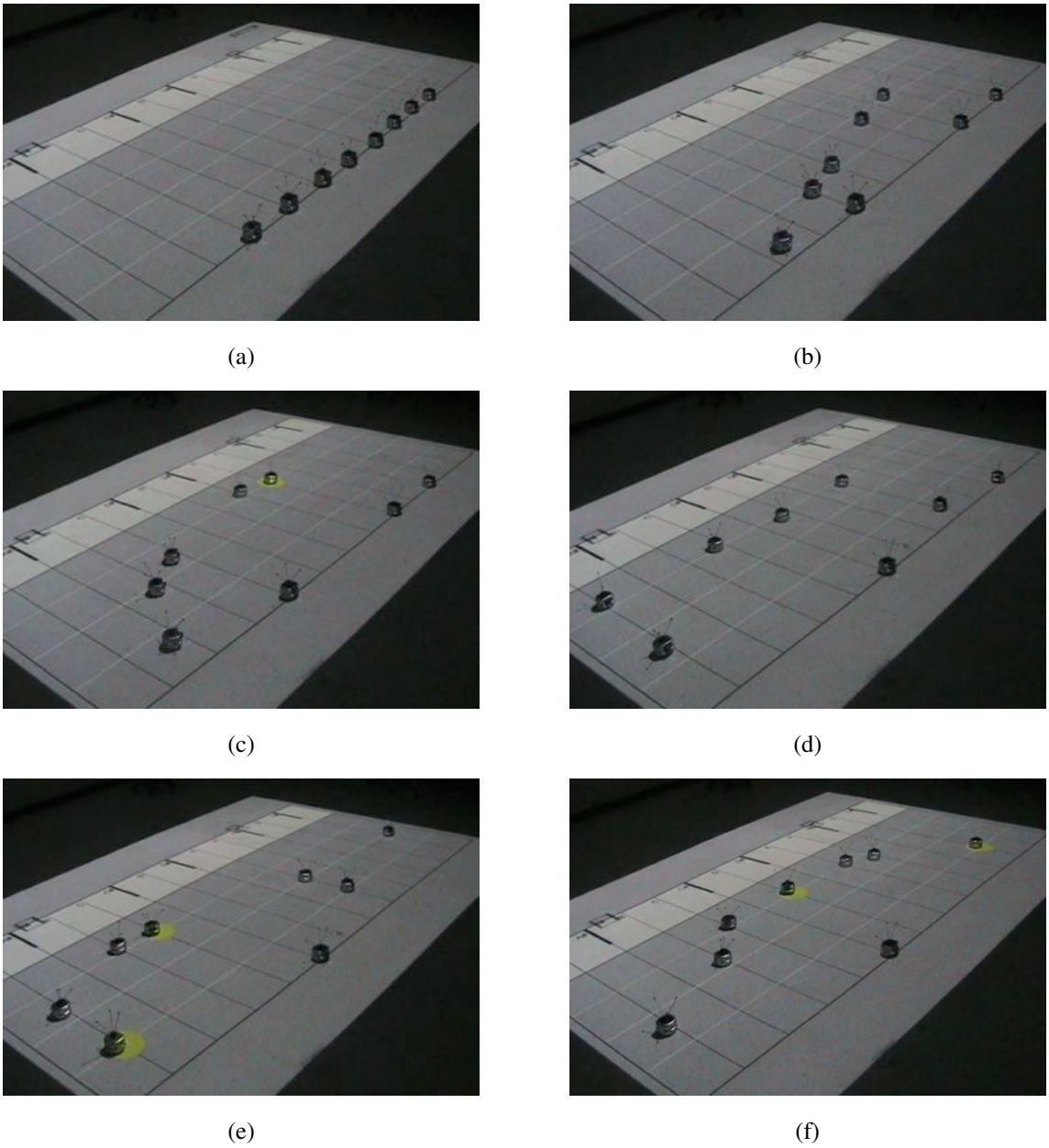
Figure 6: A line of Khepera III robots.

(blue)), from where robots could start in conjunction with Assumption 1. In our program, the instant a robot reached an assigned timed position on the wall, it was encircled by a light circle (yellow), and the sound of the corresponding piano note was generated. For the Velocity Constrained Routing Problem (VCRP), we chose different values of the maximum velocity, $\hat{v}$, and calculated the corresponding minimum number of robots $r^\star$ (see Figure 5a). Then, for different number of robots $r \geq r^\star$ (given some $\hat{v}$), we generated the optimal routes for every robot (see Figure 5b). These routes were executed by the robots with appropriate velocities that ensured their timely arrival at assigned positions.

In addition to MATLAB simulations, we conducted hardware implementations to demonstrate the VCRP. These implementations were conducted in the *Georgia Robotics and Intelligent Systems Laboratory*. The indoor facility is equipped with a motion capture system which yields real time accurate data for all tracked objects. We used Khepera III miniature robots by K Team as our hardware ground robots (see Figure 6).

By projecting the image of the Piano Wall onto the floor, we were able to demonstrate the VCRP in the musical manner similar to our MATLAB simulation. Figure 7 depicts instances of these hardware implementations. In particular, we capped the velocities of the robots at 0.15m/s, and presented them with a slow version of the Score associated with "Für Elise" (at a beat of 3 seconds). The minimum number of Khepera robots (5 in this case), found by solving the corresponding routing problem, was then deployed in order to

successfully “play” the melody (see Figures 7a - 7c). By increasing the tempo of the Score midway (3 times faster), we saw that the minimum number of required robots increased, as two more robots joined the performance (see Figures 7d - 7f). Note that we implemented low-level trajectory controllers using results from [52], in order to ensure that the robots executed their respective paths in a timely manner.



(a) (b) (c) (d) (e) (f)

Figure 7: Hardware implementation of Velocity Constrained Routing.
`http://www.youtube.com/watch?v=YigAzrFoN3E/`

3.2 Ensuring Connectivity

Often in multi-robot applications, connectivity among range constrained robots is a challenging practical concern, where in order to execute a mission in a collaborative manner, it is required that the induced information exchange network between the robots be sufficiently rich. In this chapter, we analyze the Unconstrained Routing Problem (URP) in the context of such requirements. In particular, given range constrained robots, we ensure that the underlying information exchange network induced by the positions of the robots, remains connected for all times.

Recall that for the set of robots R, we represent each robot as a planar point particle with single integrator dynamics $\dot{x}_p(t) = u_p(t)$, $p \in R$, $t \in [t_0, t_n]$. We assume that u_p is continuous almost everywhere, and we use the notation $x_p \in \hat{C}_2[t_0, t_n]$ to denote this fact (x_p denotes the differentiable *almost everywhere* trajectory of robot p over the interval $[t_0, t_n]$). Moreover, we let $X(t)$ denote the set of positions of all robots at time t, i.e., $X(t) = \{x_p(t) \,|\, p \in R\}$. By defining $\mathcal{C}_r$ as the following space,

$$\mathcal{C}_r = \underbrace{\hat{C}_2[t_0, t_n] \times ... \times \hat{C}_2[t_0, t_n]}_{r \,\text{copies}}$$

$X \in \mathcal{C}_r$ as such, denotes a collection of differentiable *almost everywhere* trajectories of the robots over the time interval $[t_0, t_n]$. Additionally, we let $d_{p,q}(t)$ denote the Euclidean distance between robots p and q, i.e., $d_{p,q}(t) = ||x_p(t) - x_q(t)||$.

Each robot has a fixed connectivity range[1] $\Delta \in \mathbb{R}$. In other words, at a given time t, robot p is connected to all robots that lie within a circle of radius Δ centered at $x_p(t)$. Since all robots possess the same range Δ, connectivity links between pairs of robots are bidirectional, i.e. if robot p is connected to robot q, then robot q is connected to robot p as well. The positions of the robots and the resulting links induce a $\Delta-$ disk proximity graph $G(X(t), \Delta)$, where the vertex set of G is given by the set R, and distinct vertices p and q

[1]The connectivity range can be a sensing footprint or a communication (broadcast) radius.

share an edge in G *if and only if* the distance between them $(d_{p,q})$ is at most equal to Δ, i.e.,

$$(p,q) \in E(G) \iff \Delta - d_{p,q} \geq 0 \tag{33}$$

where $E(G)$ denotes the edge set of G.

We are interested in the problem of routing such range constrained robots to reach timed positions in the Score, while ensuring that the underlying proximity graph induced by the positions of the robots remains connected at all times. Similar to previous cases, we view the total distance traveled as the optimization criterion. We refer to this problem as the **Connectivity Constrained Routing Problem (CCRP)** [17, 19].

3.2.1 Feasibility and Minimality

In this section, we summarize the feasibility aspects of the CCRP, and provide results on the minimum number of robots required.

Definition 5 *Given (Sc, R, P_0, Δ), $X \in \mathcal{C}_r$ is **feasible** if it satisfies the following conditions,*

1. $P_i \subseteq X(t_i)\ \forall\ i \in \{0, ..., n\}$
2. $G(X(t), \Delta)$ *is connected* $\forall\ t \in [t_0, t_n]$

The first condition ensures that every timed position in the Score is reached by a robot. The second condition ensures that the $\Delta-$ disk proximity graph induced by the positions of the robots is connected for all time over the interval $[t_0, t_n]$.

We let $\mathcal{F}_r \subseteq \mathcal{C}_r$ denote the set of all feasible sets of trajectories,

$$\mathcal{F}_r = \{X \mid X \in \mathcal{C}_r\, is\, feasible\}, \tag{34}$$

which allows us to state the Feasibility Problem : Given (Sc, Δ), the objective is to find the minimum number of robots, $r^\star$ such that $\mathcal{F}_{r^\star} \neq \emptyset$ for the corresponding $(Sc, R^\star, P_0, \Delta)$

quadruple, where $R^\star = \{1, ..., r^\star\}$ is the set of robots and P_0 is the set of their initial positions.

3.2.1.1 Establishing Feasibility:

In this section, we approach the Feasibility Problem by establishing certain conditions that ensure the *existence* of a feasible set of trajectories, enumerated in the following Theorem,

Theorem 1 *Given (Sc, R, P_0, Δ), there exists $X \in \mathcal{C}_r$ such that,*

1. $P_i \subseteq X(t_i)\ \forall\ i \in \{0, ..., n\}$
2. $G(X(t_i), \Delta)$ *is connected* $\forall\ t \in [t_0, t_n]$

if and only if there exists a set of trajectories X' such that $X' \in \mathcal{F}_r$ and $X'(t_i) = X(t_i)\ \forall\ i \in \{0, ..., n\}$.

Proof Assume that there exists $X \in \mathcal{C}_r$ that satisfies the above conditions (1) and (2). Notice that both these conditions constrain X at only paricular time instants, i.e. the initial time instant (t_0) and the subsequent Score time instants ($t_1, ..., t_n$). In other words, the conditions constrain sets of robot positions $X(t_i)$, $i \in \{0, ..., n\}$.

Consider a pair of such sets of robot positions, specified at successive time instants, denoted by say $X(t_{i-1})$ and $X(t_i)$. From condition (2), we see that the corresponding induced graphs $G(X(t_{i-1}), \Delta)$ and $G(X(t_i), \Delta)$ are connected. For such a pair of connected graphs, it was shown in [71] that there exist *connectivity preserving* motions from one configuration to another. In other words, there exists $X' \in \mathcal{C}_r$ such that $X'(t_{i-1}) = X(t_{i-1})$ and $X'(t_i) = X(t_i)$, and $G(X'(t), \Delta)$ is connected over the interval (t_{i-1}, t_i). Moreover, one can see that such a X' exists between every pair of successive sets of positions, thereby proving the existence of a set of piecewise robot trajectories $X' \in \mathcal{F}_r$, where $X'(t_i) = X(t_i)\ \forall\ i \in \{0, ..., n\}$.

Conversely, if we assume that there exists $X'(t) \in \mathcal{F}_r$ such that $X'(t_i) = X(t_i)\ \forall\ i \in \{0, ..., n\}$, then by definition, $X(t) \in \mathcal{C}_r$ satisfies conditions (1) and (2). ∎

Theorem 1 states that the positions of the robots at *particular* time instants are sufficient to determine the existence of a feasible set of trajectories. However, to ensure that the positions satisfy conditions (1) and (2), we need to first ensure that we have enough robots. The following equations establish such a requirement for a minimum number of robots,

$$r < \mathcal{K} \Rightarrow \mathcal{F}_r = \emptyset \tag{35}$$

$$r \geq \mathcal{K} \not\Rightarrow \mathcal{F}_r \neq \emptyset \tag{36}$$

Equation (35) states that if the number of robots is *less than* the maximum number of positions in the Score, that require to be reached at simultaneously, then there are not enough robots to ensure that all those positions are occupied simultaneously. In other words, condition (1) would never be satisfied, and consequently, there would exist no feasible set of trajectories. Equation (36), on the other hand, states that having a minimum of $\mathcal{K}$ number of robots *still* does not guarantee the existence of a feasible set of trajectories. For instance, it is entirely possible that, for a given range Δ, the positions requiring simultaneous reaching at some time instant in the Score are located so far apart from one another that $\mathcal{K}$ number of robots are just not enough to induce a connected $\Delta-$ disk proximity graph at that time instant. In other words, condition (2) would never be satisfied, resulting in $\mathcal{F}_r = \emptyset$.

Thus, we proceed to find the minimum number of robots $r^\star$ that ensures that conditions (1) and (2) from Theorem 1 can be met, which in turn would prove the existence of a feasible set of trajectories. To keep the problem of finding $r^\star$ independent of the initial positions of the robots, we make the following assumption,

Assumption 2 *The starting position of every robot in R is chosen such that the induced $\Delta-$ disk proximity graph $G(X(t_0), \Delta) = G(P_0, \Delta)$ is connected.*

Theorem 2 *Given a set of positions P_i specified at time t_i in the Score, and a connectivity range Δ, the problem of finding the minimum number of robots, r_i, that ensures that every*

position in P_i is occupied, and the induced $\Delta-$ disk proximity graph is connected, is equivalent to the ***Steiner tree problem with minimum number of Steiner points and bounded edge length (STP-MSPBEL)****.*

Proof The STP-MSPBEL in its general form (see [48]) is stated as follows,

"Given a set of planar positions P, and a positive constant R, the objective of the STP-MSPBEL is to find a tree spanning a superset of P such that each edge in the tree has a length no more than R and the number of points other than those in P, called Steiner points, is minimized."

We see that the problem of finding the minimum number of robots at time instant t_i is identical to the STP-MSPBEL, where the position set P_i corresponds to P and the range Δ corresponds to the positive constant R. The positions of the vertices of the solution tree denote the positions of the robots, thus ensuring that each position in P_i is occupied, and the induced $\Delta-$ disk proximity graph is connected. ∎

We denote the positions of the robots in the solution tree by S_i, and the number of *Steiner Points* by s_i. Note that $s_i + |P_i| = |S_i| = r_i$. Moreover, from Theorem 2, we get the minimum number of robots required at a particular time instant in the Score, such that conditions (1) and (2) in Theorem 1 *evaluated at that particular time instant*, can be met. However, each time instant in the Score may require a different minimum number of robots, depending on its corresponding specified position set. Thus, in order to obtain a global minimum that ensures that both conditions (1) and (2) in Theorem 1 can be met, we must take the maximum over all time instants, of the minimum number of robots.

Theorem 3 *For a given (Sc, Δ), the minimum number of robots required, $r^\star$, to ensure that $\mathcal{F}_{r^\star} \neq \emptyset$ for the corresponding $(Sc, R^\star, P_0, \Delta)$ quadruple, where $R^\star = \{1, ..., r^\star\}$ is the set of robots and P_0 is the set of their initial positions, is given by,*

$$r^\star = \min_r \{\mathcal{F}_r \neq \emptyset\} = \max\{r_i | i \in \{1, ..., n\}\} \tag{37}$$

Proof Let us assume that at time instant t_i, the minimum number of robots required is indeed the maximum over all time instants in the Score, i.e., $r_i = r^\star$. Thus, if the total number of robots r is less than $r^\star$, then at least one of the conditions (1) or (2) in Theorem 1 would never be met, thus resulting in $\mathcal{F}_r = \emptyset$.

Conversely, if the total number of robots r is at least equal to $r^\star$, then both conditions (1) or (2) in Theorem 1 can be met, thereby proving that $\mathcal{F}_r \neq \emptyset$. ∎

Calculating the minimum number of robots: From Theorem 2, it is clear that solving the STP-MSPBEL is impertinent to finding $r^\star$. However, the STP-MSPBEL is proven to be NP-hard [48]. Thus, Theorems 2 and 3 provide *theoretical* results on global optimality. To calculate $r^\star$ in practical scenarios, one can use many existing algorithms with varying time complexities and performance ratios that provide an approximation to the STP-MSPBEL [14, 15]. For instance, an $\mathcal{O}(N^3)$ time approximate algorithm with performance ratio of at most 3, is presented in [15], where N denotes the number of planar positions given in the STP-MSPBEL. Another example is the approximate algorithm obtained by the minimum spanning tree [48].

3.2.2 Generating Trajectories

In this section, we go from existence results to the generation of actual feasible trajectories. To achieve this, one can associate a cost with the trajectories, for example, the total length traveled. Or more precisely,

Given (Sc, R, P_0, Δ), where it is assumed that we have enough robots to ensure the existence of a set of feasible trajectories ($r \geq r^\star$), the objective would be to generate $X \in \mathcal{F}_r$ such that the following function is minimized,

$$\sum_{p \in R} \int_{t_0}^{t_n} \sqrt{\dot{x}_{p1}^2 + \dot{x}_{p2}^2}\, \mathrm{d}t \tag{38}$$

Due to the high dimensionality of the multi-robot configuration space, finding a global

solution i.e. a set of minimum length trajectories that are feasible, is typically not an option. As a result, we relax the requirement for global optimality and instead, propose the *Trajectories* algorithm that guarantees convergence to a sub-optimal solution.

3.2.2.1 The Trajectories Algorithm:

To generate trajectories for the CCRP, we revisit certain results regarding the Unconstrained Routing Problem (URP) in Chapter 2. Given (Sc, R, P_0, Δ), the *Trajectories* algorithm calculates the *optimal assignment* $\pi^\star$ for the corresponding URP (as discussed in 2.2), and consequently, the trajectories of every robot by linearly interpolating between successive pairs of assigned timed positions in increasing order of specified time instants. We let $X^b \in \mathcal{C}_r$ denote the set of such trajectories.

Using Δ, and the positions $X^b(t_i)$ for all $i \in \{1, ..., n\}$, as an *initial* estimate, the algorithm (inspired by Theorem 1) uses the *Connect* sub-algorithm to modify these positions in a manner that ensures that the induced proximity graph at every time instant in the Score is connected. As a result, conditions (1) and (2) of Theorem 1 are satisfied. Moreover, the algorithm uses the *Assign* sub-algorithm to reassign robots from their positions at a particular time instant to positions at the next (successive) time instant in the Score. Finally, the algorithm uses the *Mid-Config* sub-algorithm to find *connectivity preserving* motions between sets of such robot positions, specified at successive time instants, thus generating a set of feasible sub-optimal piecewise robot trajectories. In subsequent paragraphs, we provide an explanation of each sub-algorithm used in the *Trajectories* algorithm.

Assign $(X(t_{i-1}),\ S_i,\ X(t_i) \setminus P_i)$***:*** For convenience, let $A = X(t_{i-1}) = \{a_1, ..., a_{|A|}\}$, $B = S_i = \{b_1, ..., b_{|B|}\}$ and $C = X(t_i) \setminus P_i = \{c_1, ..., c_{|C|}\}$. Note that A, B and C are three sets of timed positions, where A is specified at t_{i-1} while B and C are specified at t_i, and $|A| \leq |B| + |C|$. Let the cost of assigning a position in A to a position in $B \cup C$ equal the distance between the two positions. The idea is to assign every position in A to a

Algorithm *Trajectories* (Sc, R, P_0, Δ)

$X \leftarrow X^b$, where $X^b(t_0) = P_0$ $\{X$ is initially equal to $X^b\}$
for $i = 1$ to n **do** {iterating over all time instants in the $Score$}
  if $G(X(t_i), \Delta)$ is connected **then**
    if $G(X^b(t_{i-1}), \Delta)$ is not connected **then** {initial estimates of the robot positions at t_{i-1} required modification}
      $H \leftarrow$ *Assign* $(X(t_{i-1}), X(t_i), \emptyset)$
      Using $H : X(t_{i-1}) \rightarrow X(t_i)$, update $X(t_i)$ such that the current position of robot p is given by $x_p(t_i) = H(x_p(t_{i-1})) = x_q^b(t_i)$, where $p, q \in R$ {At t_i, robot p assumes the position originally occupied by robot q}
      Update $X(t_j), j \in \{i+1, ..., n\}$ such that robot p assumes all positions originally occupied by robot q, at *all* future Score time instants, i.e. $x_p(t_j) = x_q^b(t_j)$ $\forall j \in \{i+1, ..., n\}$
    end if
  else $\{G(X(t_i), \Delta)$ is not connected$\}$
    Find S_i, i.e. the positions of r_i robots at t_i, obtained by approximately solving the STP-MSPBEL [15]
    if $S_i \neq P_i$ **then** $\{P_i$ does not induce a connected proximity graph, i.e. steiner points are added at $t_i\}$
      $H \leftarrow$ *Assign* $(X(t_{i-1}), S_i, X(t_i) \setminus P_i)$ {e.g. Figure 8}
      Using $H : X(t_{i-1}) \rightarrow S_i \cup X(t_i) \setminus P_i$, update $X(t_i)$ such that the current position of robot p is given by $x_p(t_i) = H(x_p(t_{i-1}))$
    end if
    $X(t_i) \leftarrow$ *Connect* $(S_i, X(t_i) \setminus S_i, \Delta)$ {e.g. Figure9}
    $H \leftarrow$ *Assign* $(X(t_{i-1}), X(t_i), \emptyset)$
    Using $H : X(t_{i-1}) \rightarrow X(t_i)$, update $X(t_i)$ such that the current position of robot p is given by $x_p(t_i) = H(x_p(t_{i-1}))$
  end if
  $X(t_{mid}) \leftarrow$ *Mid-Config* $(X(t_{i-1}), X(t_i), \Delta)$, $t_{mid} \in (t_{i-1}, t_i)$ {e.g. Figure 10}
  $X(t) \leftarrow$ linear interpolation between $X(t_{i-1}), X(t_{mid})$ and $X(t_i)$, $t \in (t_{i-1}, t_i)$
end for
return X

unique position in $B \cup C$, such that all positions in B are assigned, positions in C may or may not be assigned, and the total cost of assignment is minimized. In essence, the *Assign* sub-algorithm solves an unbalanced linear sum assignment problem (see [49]).

By defining $\hat{B} \triangleq \{b_1, ..., b_{|B|}, c_1, ..., c_{|C|}\} \triangleq \{\hat{b}_1, ..., \hat{b}_{|B|+|C|}\}$, we can describe the assignment problem as a linear program,

$$\min_{l} \sum_{\alpha=1}^{|A|} \sum_{\beta=1}^{|B|+|C|} ||\hat{b}_\beta - a_\alpha||\, l(\alpha, \beta) \tag{39}$$

subject to:

$$l(\alpha, \beta) \in \{0, 1\} \tag{40}$$

$$\sum_{\alpha=1}^{|A|} l(\alpha, \beta) = 1, \forall\, \beta \in \{1, ..., |B|\} \tag{41}$$

$$\sum_{\alpha=1}^{|A|} l(\alpha, \beta) \leq 1, \forall\, \beta \in \{|B|+1, ..., |B|+|C|\} \tag{42}$$

$$\sum_{\beta=1}^{|B|+|C|} l(\alpha, \beta) = 1, \forall\, \alpha \in \{1, ..., |A|\} \tag{43}$$

where $l(\alpha, \beta)$ is 1 if $a_\alpha \in A$ is assigned to $\hat{b}_\beta \in \hat{B}$, and 0 otherwise.

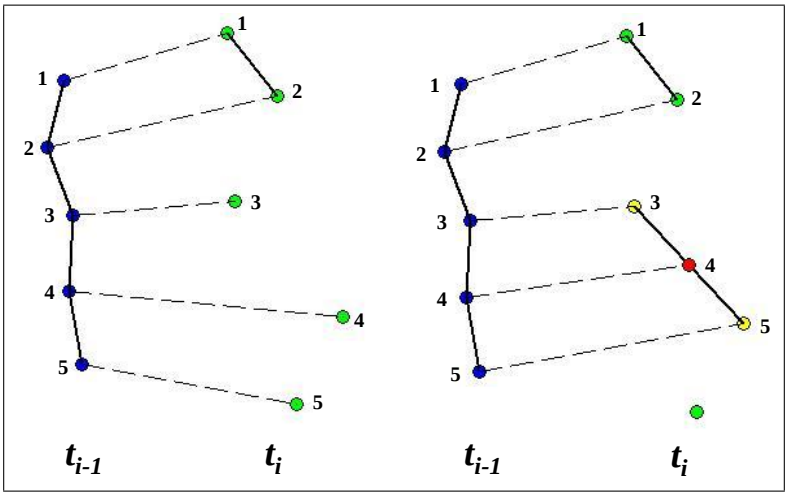


Figure 8: Left: Robot positions at t_{i-1} induce a connected graph, and at t_i, a disconnected graph, Right: Using the *Assign* sub-algorithm, robots are reassigned to positions at t_i, such that all positions in S_i are occupied (S_i contains the timed positions at t_i (two yellow circles) and the added steiner point (red circle)).

Algorithm *Assign* (A, B, C)

1: $\hat{B} \triangleq \{b_1, ..., b_{|B|}, c_1, ..., c_{|C|}\} \triangleq \{\hat{b}_1, ..., \hat{b}_{|B|+|C|}\}$
2: Find l that solves Equations (63)-(66)
3: Find $H : A \rightarrow \hat{B}$ such that $l(\alpha, \beta) = 1 \iff H(a_\alpha) = \hat{b}_\beta, \forall\, \alpha \in \{1, ..., |A|\}, \beta \in \{1, ..., |B| + |C|\}$
4: **return** H

Connect $(S_i,\, X(t_i) \setminus S_i,\, \Delta)$***:*** For convenience, let $A = S_i = \{a_1, ..., a_{|A|}\}$ and $B = X(t_i) \setminus S_i = \{b_1, ..., b_{|B|}\}$. Note that A and B are two sets of timed positions, both specified at t_i, and the induced graph $G(A, \Delta)$ is connected, i.e. positions in A form a connected backbone, while B contains positions that may or may not be connected to this backbone. The idea is to "grow" this connected backbone by recursively adding to A, updated positions from B such that the updated $G(A, \Delta)$ becomes connected. The algorithm returns this connected backbone A.

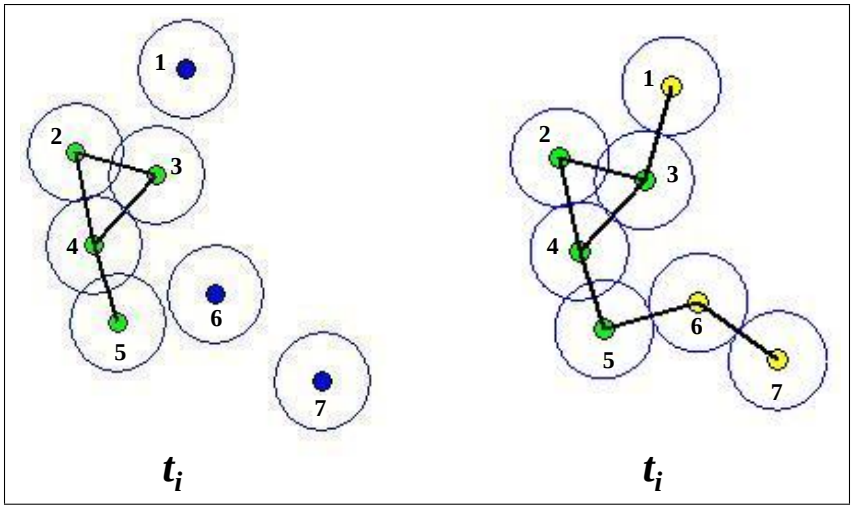


Figure 9: Left: Robots 2,3,4 and 5 form a connected backbone while robots 1,6 and 7 are disconnected, Right: Using the *Connect* sub-algorithm, robots 1,6 and 7 merge with the connected backbone (circular halos depict the range $\frac{\Delta}{2}$).

Algorithm *Connect* (A, B, Δ)

repeat
  Find $\alpha^\star \in \{1, ..., |A|\}$, $\beta^\star \in \{1, ..., |B|\}$ such that $||a_{\alpha^\star} - b_{\beta^\star}|| = \min(||a_\alpha - b_\beta||)$ $\forall \alpha, \beta$
  if $||a_{\alpha^\star} - b_{\beta^\star}|| > \Delta$ **then**
    $b_{\beta^\star} \leftarrow a_{\alpha^\star} + \frac{b_{\beta^\star} - a_{\alpha^\star}}{||a_{\alpha^\star} - b_{\beta^\star}||}\Delta$
  end if
  $A \leftarrow A \cup \{b_{\beta^\star}\}$
  $B \leftarrow B \setminus \{b_{\beta^\star}\}$
until $B = \emptyset$
return A

Mid-Config $(X(t_{i-1}),\ X(t_i),\ \Delta)$***:*** For convenience, let $A = X(t_{i-1}) = \{a_1, ..., a_{|A|}\}$ and $B = X(t_i) = \{b_1, ..., b_{|B|}\}$. Note that A and B are two sets of timed positions, specified at t_{i-1} and t_i respectively, where $|A| = |B|$. Moreover, the induced graphs $G(A, \Delta)$ and $G(B, \Delta)$ are both connected. The idea is to find an equal sized set of *intermediate* planar positions $M = \{m_1, ..., m_{|M|}\}$, specified at some time instant $t_{mid} \in (t_{i-1}, t_i)$, such that the induced proximity graph $G(M, \Delta)$ contains the edges of the *spanning trees* of *both* $G(A, \Delta)$ and $G(B, \Delta)$ (Notice that $G(M, \Delta)$ is connected by definition). Consequently, the set of piecewise linear trajectories formed by linearly interpolating between A, M and B, is guaranteed to ensure a connected proximity graph for all times $t \in (t_{i-1}, t_i)$.

Moreover, let the mid points of each unconstrained straight line path between corresponding positions a_α and b_α, $\alpha \in \{1, ..., |A|\}$ be the so-called *target* points for corresponding planar positions in M. Let $C = \{c_\alpha \,|\, c_\alpha = \frac{a_\alpha + b_\alpha}{2}, \alpha \in \{1, ..., |A|\}\}$ denote the set of these *target* points. The sub-algorithm ***Mid-Config*** then solves the following constrained optimization problem,

$$\min_M \sum_{\alpha=1}^{|A|} ||m_\alpha - c_\alpha|| \tag{44}$$

such that $G(M, \Delta)$ contains the edges of the spanning trees of $G(A, \Delta)$ and $G(B, \Delta)$.

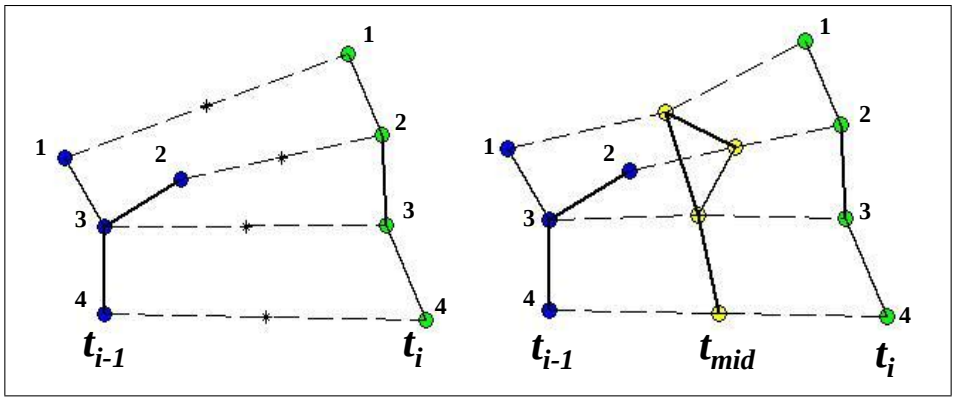


Figure 10: Left: Robots at time instants t_{i-1} and t_i induce connected graphs, Right: Using the *Mid-Config* sub-algorithm, intermediate positions of the robots are found, and the dashed lines represent the resulting connectivity preserving piecewise linear trajectories over the interval.

Algorithm *Mid-Config* (A, B, Δ)

1: $C \triangleq \{c_\alpha \,|\, c_\alpha = \frac{a_\alpha + b_\alpha}{2}, \alpha \in \{1, ..., |A|\}\}$

2: $G_s(A, \Delta) \leftarrow$ euclidean min span tree of $G(A, \Delta)$

3: $G_s(B, \Delta) \leftarrow$ euclidean min span tree of $G(B, \Delta)$

4: Find M by solving Equation 44 such that $G(M, \Delta)$ contains the edges of $G_s(A, \Delta)$ and $G_s(B, \Delta)$

5: **return** M

3.2.3 Simulation Results (Robot Music Wall 1.0)

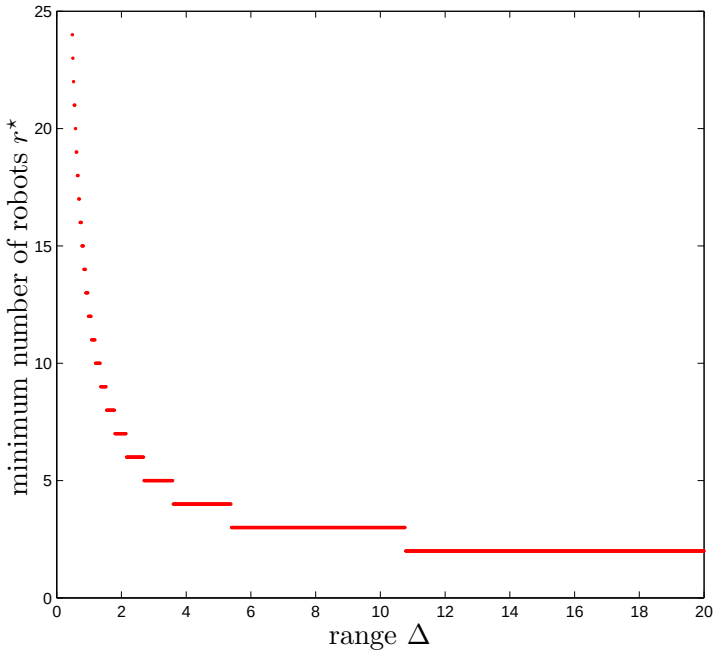
Recall the simulated Piano Wall introduced in Section 3.1.1. Similar to the implementation of the Velocity Constrained Routing Problem (VCRP) discussed in that section, our goal was to make multiple robots perform the popular composition "Für Elise" by Ludwig van Beethoven on the Piano Wall. However, in this instance, the robots were given a connectivity range Δ instead of a maximum velocity $\hat{v}$, and their trajectories were obtained by solving the corresponding CCRP.

For different values of Δ, we calculated the corresponding minimum number of robots $r^\star$ (see Figure 11a). Then, for different number of robots $r \geq r^\star$ (given some Δ), we constructed the routes for every robot, using the *Trajectories* algorithm (see Figure 11b for an example). Similar to previous implementations, such routes/paths were executed by the robots with appropriate velocities to ascertain their on-time arrival at assigned positions.

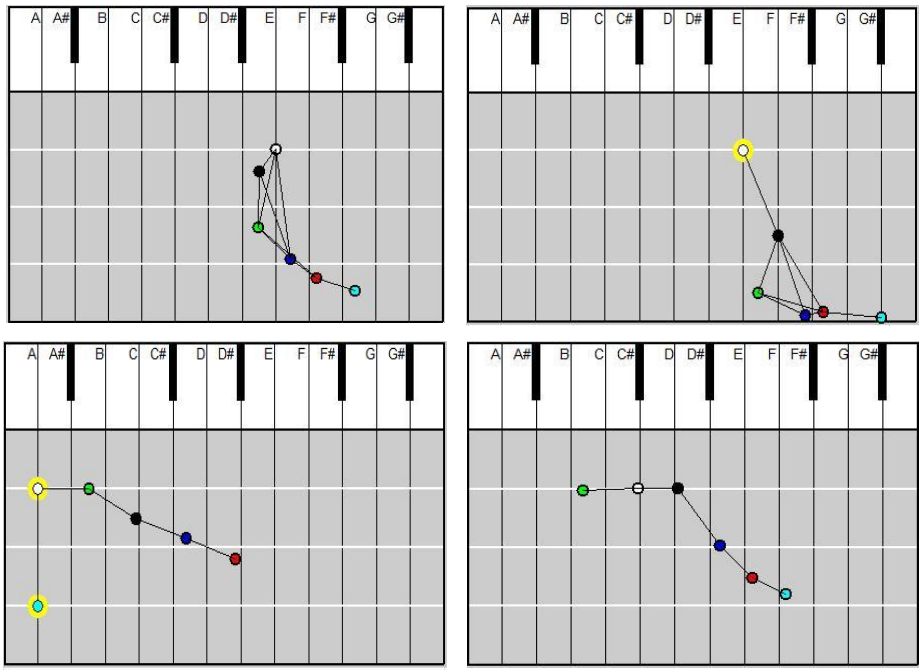
Computational Complexity: For a given (Sc, R, P_0, Δ), the *Trajectories* algorithm initially solves the Unconstrained Routing Problem (URP) by employing the Hungarian Method, the complexity of which is $O(|Sc|^3)$. Additionally, it computes piecewise linear trajectories using the various sub-algorithms, by iterating over n time instants, with the complexity of each iteration being $O(r^3)$. Thus, the overall complexity of the algorithm is $O(|Sc|^3 + nr^3)$.

3.2.3.1 Optimizing Total Length: A Discussion

In this section, we highlight various design characteristics targeted towards optimizing the total length of robot trajectories. To begin with, the *Trajectories* algorithm uses *optimal* positions of robots, obtained by solving the URP, as initial estimates for finding a sub-optimal solution to the CCRP. Moreover, the *Connect* sub-algorithm recursively moves each disconnected robot by a minimal distance, in order to merge it with a connected backbone at a particular time instant in the Score. The *Assign* sub-algorithm reassigns robots from their positions at one time instant in the Score to the next, such that the total length of the corresponding straight line robot trajectories between assigned positions is minimum, thereby providing a good base for the *Mid-Config* sub-algorithm. In turn, the *Mid-Config* sub-algorithm finds intermediate robot positions that cause a minimum deviation between the the original straight line trajectories and the resulting piecewise linear ones, while satisfying the edge constraints on the induced proximity graph.



(a) A plot of the minimum number of robots $(r^\star)$ versus the range (Δ).



(b) Four snapshots of the connectivity constrained case: $\Delta = 4$, $r^\star = 4$.

Figure 11: Simulation results for Connectivity Constrained Routing.

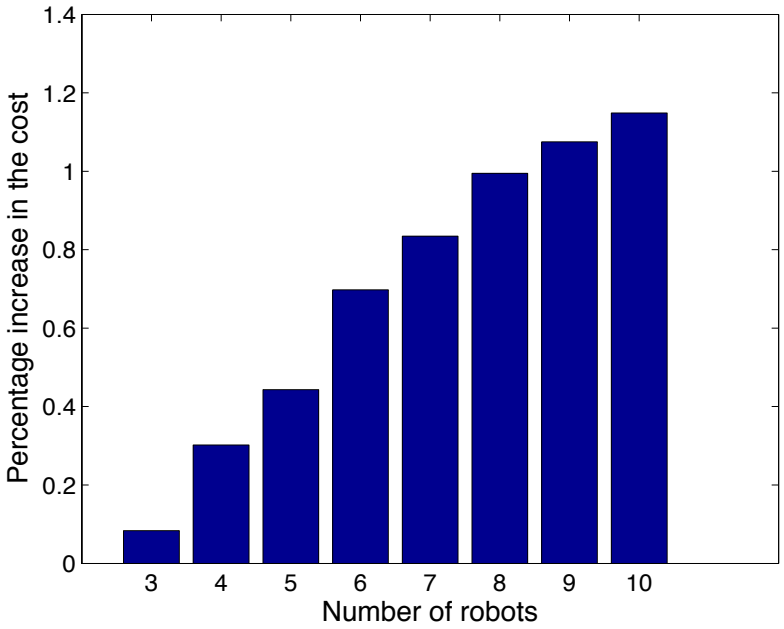


Figure 12: Average percentage increase in the cost, calculated over 5000 Scores, randomly generated each time for a different r.

As discussed before, finding an optimal solution to the CCRP is typically not feasible, since searching the space for all possible coordinated paths for multiple robots requires a prohibitive amount of computation time and memory (for instance, the general optimal motion planning problem for multiple robots is P-Space hard [36]). However, in order to characterize "how" sub-optimal the solution obtained from the *Trajectories* algorithm is, we carry out the following experiment:

For a given (Sc, R, P_0), we solve the Unconstrained Routing Problem (URP) using the Hungarian Method (see Section 2.2). Since the solution to the URP provides globally optimal robot trajectories, we calculate the minimum sensing range Δ_{free} that ensures a connected proximity graph for all times, given the robot trajectories. Thus, for the quadruple $(Sc, R, P_0, \Delta_{free})$, these trajectories provide an optimal solution to the corresponding connectivity constrained version of the problem, i.e. the CCRP. Now, by running the *Trajectories* algorithm on the same quadruple $(Sc, R, P_0, \Delta_{free})$, we obtain a sub-optimal solution to the CCRP, that can be compared to the optimal solution computed earlier.

Notice that through this approach, the optimal solution to the CCRP is the very same that is computed by the *Trajectories* algorithm on line (1), as an initialization step. Consequently, the *Trajectories* algorithm never executes lines (10) - (17), and thus, the approximation algorithm that solves the STP-MSPBEL on line (10) does not affect the optimality of the solution. In fact, the main deviation from the optimal solution is caused by the *Mid-Config* sub-algorithm, which we characterize through multiple simulation experiments. We acknowledge that this approach is targeting a simpler instance of the CCRP, but since the generalized problem is computationally intractable to solve, we hope to provide a benchmark for cases where we can in fact, find the optimal solution, and hence make a sub-optimality comparison.

In particular, for a fixed number of robots, say r, we randomly generate r number of initial and final positions from a pre-defined uniform distribution. For such a "one-beat" Score, we calculate Δ_{free} that allows the Hungarian Method to provide an optimal solution to the corresponding CCRP. Moreover, we generate the sub-optimal solution through the Trajectories algorithm, and calculate the percentage increase in the cost, i.e. the total length traveled, with respect to the optimal solution (see Figure 12). As can be seen, the increase in cost is small on the average across all cases.

CHAPTER IV

HETEROGENEOUS CAPABILITIES

Many a times, missions in multi-robot scenarios require an assortment of robotic capabilities. For instance, robots with different sensing techniques may be required to simultaneously execute different aspects of an information gathering mission. Moreover, in large scale multi-robot applications, it is usually preferable to have cheaper robots with different individual capabilities, as opposed to expensive robots enabled with all possible capabilities. As such,, we incorporate heterogeneity into the Unconstrained Routing Problem (URP) discussed in Chapter 2, by associating one or more skills (capabilities) with each robot, as well as each timed position A robot can reach a timed position only if it has *at least* one skill in common with the skill set of that position.

Recall that the Score, denoted by Sc, is the set of all timed positions that must be reached, where each timed position is expressed as a (position, time) pair (Definition 1). For convenience, let Sc_i be the set of timed positions specified at time instant $t_i \in T$, i.e.,

$$Sc_i = \{(P_{i,\alpha}, t_i) \,|\, \alpha \in \{1, ..., |P_i|\}\} \tag{45}$$

Consequently, the Score is given by $Sc = \bigcup_{i=1}^{n} Sc_i$.

We let $L = \{l_1, l_2 ..., l_{|L|}\}$ be the set of distinct skills that are required at timed positions in the Score. Moreover, we let $M_{pos} : Sc \rightarrow 2^L \setminus \emptyset$ describe a mapping between timed positions and skill sets, such that $M_{pos}((P_{i,\alpha}, t_i) \in Sc) \subseteq L \neq \emptyset$ gives the skill set associated with the timed position $(P_{i,\alpha}, t_i) \in Sc$.

Recall that $R = \{1, 2, ..., r\}$ denotes the set of r robots, and $P_0 = \{P_{0,\alpha} \,|\, \alpha \in \{1, ..., r\}\}$ denotes the set of their initial positions, defined at some initial time instant t_0. Similar to how every timed position in the Score has an associated skill set, every robot in R has an

associated skill set, described through the function $M_{rbt} : R \rightarrow 2^L \setminus \emptyset$.

We are interested in the problem of routing these robots to reach the timed positions contained in the Score, under the condition that a robot can be assigned to a timed position only if it has a skill in common with the associated skill set of that timed position. We refer to this problem as the **Heterogeneous Routing Problem (HRP)** [18].

4.1 Feasible Routes

In this section, we discuss the feasibility aspects of the HRP, i.e. if it is even possible to execute the Score with a given set of resources. Additionally, we provide an algorithm for calculating the minimum number of robots required.

Definition 6 *The quintuple* $(Sc, R, L, M_{pos}, M_{rbt})$ *is* ***feasible*** *if there exists a mapping* $A : R \rightarrow 2^{Sc}$ *between robots in* R *and sets of timed positions in the Score, that satisfies the following conditions,*

$$\bigcup_{p \in R} A(p) = Sc \tag{46}$$

$$A(p) \cap A(q) = \emptyset \quad \forall\, p, q \in R, p \neq q \tag{47}$$

$$(P_{i,\alpha}, t_i), (P_{j,\beta}, t_j) \in A(p) \Rightarrow i \neq j \quad if\ \alpha \neq \beta \tag{48}$$

$$(P_{i,\alpha}, t_i) \in A(p) \Rightarrow M_{pos}((P_{i,\alpha}, t_i)) \cap M_{rbt}(p) \neq \emptyset \tag{49}$$

Equation (46) states that every timed position in the Score is assigned, while Equation (47) states that no two robots are assigned the same timed position. Equation (48) states that a robot is not assigned more than one position at a given time instant, and Equation (49) states that if a robot is assigned to a timed position, then it must have at least one skill in common with the associated skill set of that timed position. Note that for such a mapping, the path of every robot can be determined by its assigned set of timed positions, traversed in increasing order of specified time instants.

4.1.1 Existence Results

For a given $(Sc, R, L, M_{pos}, M_{rbt})$, we would like to characterize the conditions that (R, M_{rbt}) must satisfy, such that $(Sc, R, L, M_{pos}, M_{rbt})$ is feasible. In other words, what are the conditions on robots and their respective skill sets, that ensure the *existence* of a mapping $A : R \to 2^{Sc}$ as per Definition 6.

We let $S_{grp} = \{s_1, s_2, ..., s_{|S_{grp}|}\}$ denote the set of all distinct skill sets associated with the robots in R, i.e. $S_{grp} = Range(M_{rbt})$. Moreover, we let $R_{grp} = \{r_1, r_2, ..., r_{|S_{grp}|}\}$ denote the set of identically skilled *groups* of robots, where $r_j \in R_{grp} = |\{p \in R \,|\, M_{rbt}(p) = s_j\}|$ i.e., r_j is the number of robots with skill set $s_j \in S_{grp}$. The total number of robots, $r = \sum r_j, \forall r_j \in R_{grp}$.

Note that (S_{grp}, R_{grp}), though constructed using (R, M_{rbt}), does not retain the information on individual robots in R and their corresponding skill sets described through M_{rbt}. Instead, it simply enumerates the distinct skill sets *available* for use, and the *number* of robots per set. However, not only is the information in (S_{grp}, R_{grp}) sufficient to characterize feasibility, it is also convenient for formulating the optimization problem for finding the minimum number of robots, as we will see later in the paper. In the following Lemma, we provide the conditions on the robots and their respective skill sets, that ensure feasibility. Note that we express these conditions on (S_{grp}, R_{grp}), constructed using (R, M_{rbt}).

Lemma 2 *Given $(Sc, R, L, M_{pos}, M_{rbt})$, for every time $t_i \in T$, there exists a function $\Pi_i : S_{grp} \to 2^{Sc_i}$ such that,*

$$\bigcup_{s_j \in S_{grp}} \Pi_i(s_j) = Sc_i \tag{50}$$

$$\Pi_i(s_j) \cap \Pi_i(s_k) = \emptyset \quad \forall\, s_j, s_k \in S_{grp}, s_j \neq s_k \tag{51}$$

$$s_j \cap M_{pos}((P_{i,\alpha}, t_i)) \neq \emptyset \quad \forall (P_{i,\alpha}, t_i) \in \Pi_i(s_j) \tag{52}$$

$$|\Pi_i(s_j)| \leq r_j \in R_{rgp} \quad \forall s_j \in S_{grp} \tag{53}$$

if and only if $(Sc, R, L, M_{pos}, M_{rbt})$ is feasible.

Proof Assume that for every time $t_i \in T$, there exists a function $\Pi_i : S_{grp} \rightarrow 2^{Sc_i}$ such that Equations (50) - (53) are satisfied. Note that Π_i is a function that maps each skill set to one or more timed positions specified at t_i. In other words, for skill set $s_j \in S_{grp}$, $\Pi_i(s_j)$ denotes the timed positions that are assigned robots with skill set s_j. From Equation (52), we see that each timed position contained in $\Pi_i(s_j)$ has a skill in common with s_j. Moreover, Equation (53) states that the total number of robots with skill set s_j, assigned to timed positions at t_i, given by $|\Pi_i(s_j)|$, does not exceed r_j, i.e. total number of *available* robots with skill set s_j. Equations (50) and (51) further state that *every* timed position at t_i is assigned to some skill set (or equivalently some robot with that skill set), and no two skill sets (or robots) are assigned to the same timed position.

Thus, the set of Π_is implies that at every time $t_i \in T$, each timed position specified at t_i can be assigned a robot in R (where each timed position is assigned to exactly one robot, each robot is assigned to at most one timed position, and an assignment implies that the robot has *at least* one skill in common with the skill set of that timed position). Consequently, we can construct a mapping $A : R \rightarrow 2^{Sc}$ by combining the above mentioned robot assignments to timed positions, over *all* time instants in the Score. Moreover, A satisfies the conditions in Definition 6, further implying that $(Sc, R, L, M_{pos}, M_{rbt})$ is feasible (Note that for a given set of Π_is, the mapping A need not be unique).

Conversely, if we assume that $(Sc, R, L, M_{pos}, M_{rbt})$ is feasible, then there exists a mapping $A : R \rightarrow 2^{Sc}$ as per Definition 6. For such a mapping A, by definition, there exist Π_is that satisfy Equations (50) - (53) (Note that for a given A, the set of Π_is is unique). ■

For convenience, we let $\Pi = \{\Pi_i \,|\, i \in \{1, ..., n\}\}$ denote the set of Π_is that satisfy Equations (50) - (53) of Lemma 2.

4.1.1.1 Minimum Number of Robots

Once we establish feasibility, we can go one step further and optimize the total number of robots required. More formally, we state the Minimality Problem:

Given a quintuple $(Sc, R, L, M_{pos}, M_{rbt})$ that is feasible, the objective is to find the minimum number of robots, $r^\star$, such that feasibility is ensured, i.e.,

1. there exists some $R' \subseteq R$, $|R'| = r^\star$, such that $(Sc, R', L, M_{pos}, M_{rbt})$ is feasible
2. there exists no $R^\star \subset R$, $|R^\star| < r^\star$, such that $(Sc, R^\star, L, M_{pos}, M_{rbt})$ is feasible

In the remainder of this section, we pose the Minimality problem of finding $r^\star$ as a combinatorial-optimization assignment problem, and provide the ***MinBots*** algorithm for solving it.

Assuming $(Sc, R, L, M_{pos}, M_{rbt})$ is feasible, we know from Lemma 2 that at every time $t_i \in T$, there exists a function $\Pi_i : S_{grp} \rightarrow 2^{Sc_i}$ such that Equations (50) - (53) are satisfied. Recall that for such a set of Π_is, denoted by Π, the number of robots required with skill set $s_j \in S_{grp}$, at a particular time $t_i \in T$, is given by $|\Pi_i(s_j)|$. We denote this number by $r^{\Pi}_{i,j}$. Consequently, the total number of robots required with skill set $s_j \in S_{grp}$, over *all* time instants, is given by $\max_{i \in \{1,\ldots,n\}} r^{\Pi}_{i,j}$. We denote this number by r^{Π}_j. For convenience, we let $R^{\Pi}_{grp} = \{r^{\Pi}_j \mid j \in \mathcal{J}\}$. Finally, the total number of robots required over all time instants *and* all skill sets, denoted by r_Π, is given by,

$$r^{\Pi} = \sum_{r^{\Pi}_j \in R^{\Pi}_{grp}} r^{\Pi}_j \quad \leq r \tag{54}$$

where we reiterate that r_Π is the total number of robots required, *given* a particular Π.

Consequently, we can express the problem of finding $r^\star$ as an optimization problem that searches for a Π, i.e. a set of Π_is that satisfy Equations (50) - (53) of Lemma 2, thereby ensuring feasibility, while minimizing the total number of robots required, r_Π, given by Equation (54).

To do so, we define a $0-1$ element cost matrix $C = [c(j, i, \alpha)]$ of size $|S_{grp}| \times |Sc|$, where $c(j, i, \alpha) = 1$ *if and only if* the skill set $s_j \in S_{grp}$ contains a skill in common with

the skill set of the timed position $(P_{i,\alpha}, t_i) \in Sc$.

For convenience, we let $\mathcal{I} \triangleq \{1, ..., n\}$ be the index set for i, representing the time indices at which positions are specified in Sc, and $\mathcal{J} \triangleq \{1, ..., |S_{grp}|\}$ be the index set for j, representing the skill set indices in S_{grp}. By defining the mapping $l(i, \alpha, j)$, we state the optimization problem as follows,

$$\min_{l} \quad \sum_{j \in \mathcal{J}} \left(\max_{i \in \mathcal{I}} \sum_{\alpha \in \mathcal{A}_i} c(j, i, \alpha)\, l(j, i, \alpha) \right) \tag{55}$$

such that,

$$l(j, i, \alpha) \in \{0, 1\} \tag{56}$$

$$\sum_{j \in \mathcal{J}} c(j, i, \alpha)\, l(j, i, \alpha) = 1 \quad \forall i \in \mathcal{I}, \alpha \in \mathcal{A}_i \tag{57}$$

$$\max_{i \in \mathcal{I}} \sum_{\alpha \in \mathcal{A}_i} c(j, i, \alpha)\, l(j, i, \alpha) \leq r_j \quad \forall j \in \mathcal{J} \tag{58}$$

where $l(j, i, \alpha)$ represents the assignment of a particular skill set $s_j \in S_{grp}$ to a timed position $(P_{i,\alpha}, t_i) \in Sc$, and is 1 if the assignment is done, and 0 otherwise. Using l, we can construct $\Pi_i : S_{grp} \rightarrow 2^{Sc_i}$ at every time $t_i \in T$, where $c(j, i, \alpha) l(j, i, \alpha) = 1 \iff (P_{i,\alpha}, t_i) \in \Pi_i(s_j)$. Note that Equations (57) and (58) ensure that Π_i satisfies Equations (50) - (53) of Lemma 2, for every time $t_i \in T$, thereby ensuring feasibility, while (55) minimizes the total number of robots required.

Conversion to a Linear Program: Note that the above mentioned optimization problem can be converted into the following linear program, and can be solved using one of many techniques available for solving linear programs (see [67]).

$$\min_{l} \quad \sum_{j \in \mathcal{J}} u_j$$

such that,

$$l(j,i,\alpha) \in \{0,1\}$$

$$\sum_{j \in \mathcal{J}} c(j,i,\alpha)\, l(j,i,\alpha) = 1 \quad \forall i \in \mathcal{I}, \alpha \in \mathcal{A}_i$$

$$u_j \geq \sum_{\alpha \in \mathcal{A}_i} c(j,i,\alpha)\, l(j,i,\alpha) \quad \forall i \in \mathcal{I}, j \in \mathcal{J}$$

$$u_j \leq r_j \quad \forall j \in \mathcal{J}$$

However, in keeping with the theme of assignment problems that runs throughout the work presented in this dissertation, we provide the *MinBots* Algorithm in the subsequent section, to optimize the number of robots required, by iteratively solving modified versions of the Linear Sum Assignment Problem [55].

The *MinBots* Algorithm: In this section, we provide a description of the *MinBots* algorithm (Algorithm), and the *ReduceCost* sub-algorithm, developed for solving the assignment problem described in Equations (55)-(58). The main idea behind the *MinBots* algorithm is as follows: For a given $(Sc, R, L, M_{pos}, M_{rbt})$ that is feasible, the algorithm finds Π[1] (set of Π_is that satisfy Equations (50) - (53) of Lemma 2). Given Π, the algorithm calculates the cost to be minimized, i.e. the total number of robots required, r^{Π}, using Equation (54). Beyond this point, the objective of the algorithm is to reduce this cost, by updating individual Π_is in a systematic manner, using the *ReduceCost* sub-algorithm, until convergence is achieved. We elaborate on this in subsequent paragraphs.

The *MinBots* algorithm evaluates every skill set in S_{grp}, one at at time, as follows: For $s_{j^\star} \in S_{grp}$[2], the *MinBots* algorithm finds all time instants t_i such that $r^{\Pi}_{i,j^\star} = r^{\Pi}_{j^\star}$, i.e. the total number of robots with skill set $s_{j^\star}$, required at t_i, is equal to the total number of robots with skill set $s_{j^\star}$, required over *all* time instants in the Score. We let $\mathcal{I}' = \{i \in \mathcal{I} \,|\, r^{\Pi}_{i,j^\star} =$

[1]Equivalent to finding an l that satisfies Equations (56) to (58), since we can construct Π from such an l.

[2]The order in which skill sets are chosen is not pertinent to finding the minimum number of robots required.

Algorithm *MinBots* $(Sc, R, L, M_{pos}, M_{rbt})$

1: $(S_{grp}, R_{grp}) \leftarrow (R, M_{rbt})$

2: $\mathcal{I}, \mathcal{I}' \leftarrow \{1, ..., n\}$; $\mathcal{J}, \mathcal{J}' \leftarrow \{1, ..., |S_{grp}|\}$

3: $R^{\Pi}_{grp} \leftarrow R_{grp}$ {Initialize R^{Π}_{grp} to R_{grp}, where $r^{\Pi}_{j} \in R'_{grp}$ represents the number of robots required, with skill set $s_j \in S_{grp}$}

4: For each $i \in \mathcal{I}$, find some initial $\Pi_i : S_{grp} \rightarrow 2^{Sc_i}$ that satisfies Equations (50) - (53) of Lemma 2 {The set of Π_is is denoted by Π}

5: Given Π from line 4, update each $r^{\Pi}_{j} \in R^{\Pi}_{grp}$ as follows: $r^{\Pi}_{j} \leftarrow \max_{i \in \mathcal{I}} r^{\Pi}_{i,j}$

6: $r^{\Pi} \leftarrow \sum_{j \in \mathcal{J}} r^{\Pi}_{j}$ {r^{Π} denotes the total number of robots required}

7: $r^{\Pi}_{pr} \leftarrow 0$ {Initialize variable to 0}

8: **while** $\mathcal{J}' \neq \emptyset$ **do** {There exist unevaluated skill sets in S_{grp}, with corresponding indices in $\mathcal{J}'$}

9: Choose $j^{\star} \in \mathcal{J}'$ {Corresponding skill set $s_{j^{\star}} \in S_{grp}$ is chosen for evaluation}

10: **while** $r^{\Pi}_{pr} \neq r^{\Pi}$ **do** {Cost has not converged}

11: $r^{\Pi}_{pr} \leftarrow r^{\Pi}$

12: $\mathcal{I}' \leftarrow \{i \in \mathcal{I} \,|\, r^{\Pi}_{i,j^{\star}} = r^{\Pi}_{j^{\star}}\}$ {$\mathcal{I}'$ contains all time indices at which the number of robots required with skill set $s_{j^{\star}}$ equals $r^{\Pi}_{j^{\star}}$}

13: **while** $\mathcal{I}' \neq \emptyset$ **do** {There exist unevaluated Π_is with corresponding time indices in $\mathcal{I}'$}

14: Choose $i \in \mathcal{I}'$ {Corresponding Π_i is chosen for evaluation}

15: $\Pi_i \leftarrow$ ***ReduceCost*** $(Sc_i, M_{pos}, R^{\Pi}_{grp}, S_{grp}, j^{\star})$

16: $\mathcal{I}' \leftarrow \mathcal{I}' \setminus \{i\}$ {Π_i is updated, thus remove corresponding time index from evaluation set}

17: **end while**

18: Update $r^{\Pi}_{j^{\star}} \in R^{\Pi}_{grp}$ as follows: $r^{\Pi}_{j^{\star}} \leftarrow \max_{i \in \mathcal{I}} r^{\Pi}_{i,j^{\star}}$

19: Update r^{Π} as follows: $r_{\Pi} \leftarrow \sum_{j \in \mathcal{J}} r^{\Pi}_{j}$

20: **end while**

21: $\mathcal{J}' \leftarrow \mathcal{J}' \setminus \{j^{\star}\}$ {Skill set $s_{j^{\star}} \in S_{grp}$ is evaluated, thus remove corresponding index from evaluation set}

22: **end while**

23: $r^{\star} \leftarrow r^{\Pi}$ {$r^{\star}$ is the minimum number of robots}

24: **return** Π, $r^{\star}$ {Π solves Equations (55)-(58)}

$r^{\Pi}_{j^\star}\}$ denote the set of indices corresponding to such time instants. Moreover, at *each* such time instant, t_i, the algorithm calls upon the *ReduceCost* sub-algorithm, that updates the corresponding individual function Π_i.

ReduceCost **(**$Sc_i, M_{pos}, R^{\Pi}_{grp}, S_{grp}, j^\star$**):** The objective of the sub-algorithm is to find $\hat{\Pi}_i$ such that $r^{\hat{\Pi}}_{i,j^\star}$ is minimized, while ensuring that $r^{\hat{\Pi}}_{i,j} \leq r^{\Pi}_{j}$ for all $j \in \mathcal{J} \setminus \{j^\star\}$. In other words, the *ReduceCost* sub-algorithm minimizes the total number of robots with skill set $s_{j^\star}$, required at t_i, while ensuring that the total number of robots with skill set $s_j \neq s_{j^\star}$, required at t_i, *does not exceed* r^{Π}_{j}. Note that to maintain feasibility, $\hat{\Pi}_i$ must satisfy Equations (50) - (53) of Lemma 2.

Thus, for a $0-1$ element cost matrix $C = [c(j,\alpha)]$ of size $|S_{grp}| \times |Sc_i|$, where $c(j,\alpha) = 1$ *if and only if* the skill set $s_j \in S_{grp}$ contains a skill in common with the skill set of the timed position $(P_{i,\alpha}, t_i) \in Sc_i$, the *ReduceCost* sub-algorithm solves the following assignment problem[3]:

$$\min_{l} \quad \sum_{\alpha \in \mathcal{A}_i} c(j^\star,\alpha)\, l(j^\star,\alpha) \tag{59}$$

such that,

$$l(j,\alpha) \in \{0,1\} \tag{60}$$

$$\sum_{j \in \mathcal{J}} c(j,\alpha)\, l(j,\alpha) = 1 \quad \forall \alpha \in \mathcal{A}_i \tag{61}$$

$$\sum_{\alpha \in \mathcal{A}_i} c(j,\alpha)\, l(j,\alpha) \leq r^{\Pi}_{j} \quad \forall j \in \mathcal{J} \tag{62}$$

where $l(j,\alpha)$ represents the assignment of a particular skill set $s_j \in S_{grp}$ to a timed position $(P_{i,\alpha}, t_i) \in Sc_i$, and is 1 if the assignment is done, and 0 otherwise. Consequently, $\hat{\Pi}_i : S_{grp} \rightarrow 2^{Sc_i}$ can be constructed using l, where $c(j,\alpha)\, l(j,\alpha) = 1 \iff (P_{i,\alpha}, t_i) \in \hat{\Pi}_i(s_j)$.

Note 1: Since the *ReduceCost* sub-algorithm is applied at all time instants $t_i \in T$ where $r^{\Pi}_{i,j^\star} = r^{\Pi}_{j^\star}$, the result is a set of corresponding $\hat{\Pi}_i$s. However, for the remaining time

[3]The assignment problem is feasible, since Π_i satisfies Equations (50) - (53) of Lemma 2.

instants, t_i, at which the *ReduceCost* sub-algorithm is *not* applied, we let $\hat{\Pi}_i = \Pi_i$, i.e. $\hat{\Pi}_i = \Pi_i \quad \forall i \in \mathcal{I} \setminus \mathcal{I}'$. For convenience, we let $\hat{\Pi} = \{\hat{\Pi}_i \,|\, i \in \mathcal{I}\}$ denote the set of all $\hat{\Pi}_i$s.

Lemma 3 *Given Π, and a skill set $s_{j^\star} \in S_{grp}$, if we apply the ReduceCost sub-algorithm at all time instants t_i where $r^{\Pi}_{i,j^\star} = r^{\Pi}_{j^\star}$, the resulting $\hat{\Pi}$ (constructed as per Note 1) satisfies the following: $r^{\hat{\Pi}} \leq r^{\Pi}$.*

Proof The proof follows directly from Equation (62) which ensures that for all $j \in \mathcal{J}$, $r^{\hat{\Pi}}_j \leq r^{\Pi}_j$. In other words, the *ReduceCost* sub-algorithm ensures that for all skill sets $s_j \in S_{grp}$, the total number of robots with skill set s_j, required over all time instants in the Score, does not increase. As a consequence, the total number of robots required over all time instants *and* all skill sets does not increase, i.e. $r^{\hat{\Pi}} \leq r^{\Pi}$. ∎

Theorem 4 *Given a quintuple $(Sc, R, L, M_{pos}, M_{rbt})$ that is feasible as per Definition 6, the MinBots algorithm converges to the minimum number of robots required, given by $r^\star$, such that feasibility is ensured, i.e.,*

(a) there exists some $R' \subseteq R$, $|R'| = r^\star$, such that $(Sc, R', L, M_{pos}, M_{rbt})$ is feasible

(b) there exists no $R^\star \subset R$, $|R^\star| < r^\star$, such that $(Sc, R^\star, L, M_{pos}, M_{rbt})$ is feasible

Proof Note that at the termination of the *MinBots* algorithm, the total number of robots required, $r^{\Pi} = r^\star$, is calculated with respect to a particular Π, i.e. a set of Π_is that satisfy Equations (50) - (53) of Lemma 2. Hence, using Lemma 2, we can conclude that there exists some $R' \subseteq R$, $|R'| = r^\star$, such that $(Sc, R', L, M_{pos}, M_{rbt})$ is feasible or in other words, condition (a) of Theorem 4 is satisfied.

For condition (b) of Theorem 4, we provide the following proof by contradiction: Let us assume that condition (b) is not satisfied. In other words, $r^\star \neq r_{min}$, where r_{min} denotes the minimum number of robots required. Since $(Sc, R, L, M_{pos}, M_{rbt})$ is feasible, we know

that $r^\star \not< r_{min}$. Moreover, $r^\star > r_{min}$ implies that for the given Π, there exists at least one skill set $s_{j^\star} \in S_{grp}$ such that $r^\Pi_{j^\star}$, i.e. the total number of robots required with skill set $s_{j^\star}$, over all time instants in the Score, can be reduced. However, from Lemma 3, we can see that for a given skill set $s_{j^\star}$, the while loop on line (10) always terminates with a reduction in $r^\Pi_{j^\star}$ and consequently a reduction in the cost, r^Π, whenever a reduction is possible. Since the *MinBots* algorithm evaluates *all* skill sets in S_{grp}, it follows that there exists no skill set $s_{j^\star} \in S_{grp}$ such that $r^\Pi_{j^\star}$ can be reduced. In other words, there exists no $R^\star \subset R$, $|R^\star| < r^\star$, such that $(Sc, R^\star, L, M_{pos}, M_{rbt})$ is feasible, i.e. condition (b) of Theorem 4 is satisfied. ■

Computational Complexity: The *ReduceCost* sub-algorithm can be solved using the Hungarian Method, and thus, has a complexity of $\mathcal{O}((r^\Pi)^3)$, where $r^\Pi \leq r$. Moreover, for each skill in S_{grp}, the sub-algorithm is called n times, where n is the total number of time instants in the Score. By bounding the number of *available* robots per skill set, $r_j \in R_{grp}$, to be no more that $\mathcal{K}$[4], where $\mathcal{K}$ denotes the maximum number of timed positions that must be reached simultaneously, we provide the following upper bound on the total number of robots, $r = \sum_{r_j \in R_{grp}} r_j \leq |S_{grp}|\mathcal{K}$. Thus, the computational complexity of the *MinBots* algorithm is given by $\mathcal{O}(nr^3|S_{grp}|)$, where $r \leq |S_{grp}|\mathcal{K}$.

4.2 Generating Routes

Up until this point, we have discussed feasibility and minimality aspects of the routing problem, in that under what conditions on a given set of resources (robots) is it possible to execute a Score, and how can we optimize these resources. However, we have not dealt with methods that translate to actual robotic motion. In this section, we provide one such method through the *PathGen* algorithm, that generates explicit paths for the robots, required to execute a Score.

[4]This bound has no implications on the calculation of $r^\star$.

In essence, for a quintuple $(Sc, R, L, M_{pos}, M_{rbt})$ that is feasible, the ***PathGen*** algorithm finds a mapping $A : R \to 2^{Sc}$ as per Definition 6, by iteratively assigning robots to timed positions specified at a particular time instant.

Algorithm *PathGen* $(Sc, R, L, P_0, M_{pos}, M_{rbt})$

1: Define $A : R \to 2^{Sc}$, and initialize as follows: $A(p) = \emptyset \quad \forall p \in R$
2: Define $P_{cur} : R \to \mathbb{R}^2$, where $P_{cur}(p)$ denotes the planar position that robot p occupies, and initialize as follows: $P_{cur}(p) \leftarrow (P_{0,p}) \quad \forall p \in R$
3: **for** $i = 1$ to n **do** {iterating over all time instants in the $Score$}
4: $H^\star \leftarrow$ ***Assign2*** $(Sc_i, R, P_{cur}, M_{pos}, M_{rbt})$ {Find $H^\star : R' \to Sc_i$, $R' \subseteq R$, that encodes the new positions occupied by all robots in R'}
5: Using $H^\star : R' \to Sc_i$, $R' \subseteq R$, update A to include the new positions occupied by robots in R', i.e. $\forall p \in R'$, $A(p) \leftarrow A(p) \cup H^\star(p)$
6: Update P_{cur}, i.e. $\forall p \in R'$, $P_{cur}(p) \leftarrow P_{i,\alpha}$, where $(P_{i,\alpha}, t_i) \in H^\star(p)$
7: **end for**
8: **return** A

Assign2 **(**$Sc_i, R, P_{cur}, M_{pos}, M_{rbt}$**):** The main idea behind the *Assign* sub-algorithm is to assign robots to timed positions at t_i where the cost of assigning a robot to a timed position is the distance between the robot's current position and that timed position. Each timed position is assigned to exactly one robot, each robot is assigned to at most one timed position, and an assignment is made *only* if the robot has a skill in common with the skill set of that timed position.

More formally, the *Assign* sub-algorithm finds some $R' \subseteq R$ such that firstly, the restricted function $H|_{R'} R' \to Sc_i$ is a bijection, where $H(p \in R) = (P_{i,\alpha}, t_i) \in Sc_i \Rightarrow M_{rbt}(p) \cap M_{pos}((P_{i,\alpha}, t_i)) \neq \emptyset$, and secondly, the total cost of the assignment is minimum. We let $H^\star$ denote such a function. Thus, the *Assign* sub-algorithm essentially solves an unbalanced linear sum assignment problem (see [21]) described as follows,

$$\min_{l} \sum_{p \in R} \sum_{\alpha \in \mathcal{A}_i} ||P_{i,\alpha} - P_{cur}(p)||\, l(p, \alpha) \tag{63}$$

subject to:

$$l(p,\alpha) \in \{0,1\} \tag{64}$$

$$\sum_{p \in R} l(p,\alpha) = 1, \quad \forall\, \alpha \in \mathcal{A}_i \tag{65}$$

$$\sum_{\alpha \in \mathcal{A}_i} l(p,\alpha) \leq 1, \quad \forall\, p \in R \tag{66}$$

$$l(p,\alpha) = 1 \Rightarrow M_{rbt}(p) \cap M_{pos}((P_{i,\alpha}, t_i)) \neq \emptyset \tag{67}$$

where $l(p,\alpha)$ represents the individual assignment of $p \in R$ to $(P_{i,\alpha}, t_i) \in Sc_i$, and is 1 if the assignment is done, and 0 otherwise. The resulting l gives us $H^\star : R' \rightarrow Sc_i$, where $l(p,\alpha) = 1 \iff H^\star(p \in R') = (P_{i,\alpha}, t_i) \in Sc_i$.

By construction, we see that the *PathGen* algorithm terminates with a mapping $A : R \rightarrow 2^{Sc}$ that satisfies Equations (46) - (49) of Definition 6. Moreover, the path of every robot can be constructed by traversing its assigned set of timed positions in increasing order of specified time instants.

Computational Complexity: The assignment problem, described in the *Assign* sub-algorithm, can be solved using the Hungarian Method, with complexity $\mathcal{O}(r^3)$ and thus, the computational complexity of the *PathGen* algorithm is given by $\mathcal{O}(nr^3)$, where $r \leq |Sc|$.

4.3 Simulation Results (Robot Music Wall 2.0)

In this section, we introduce the second simulated (heterogeneous) version of the Robot Music Wall, developed in MATLAB and instrumented to include piano, guitar and drum sounds (see Figure 13). This time around, we chose to demonstrate the Heterogeneous Routing Problem (HRP) by making multiple robots (each robot could play one or more instruments) perform an instrumental version of the song "Can You Feel the Love Tonight" by Elton John (from the movie The Lion King). For the purpose of exhibiting feasibility and minimality results with clarity and flexibility, we created a graphical user-interface (GUI) around the Robot Music Wall 2.0, also depicted in Figure 13.

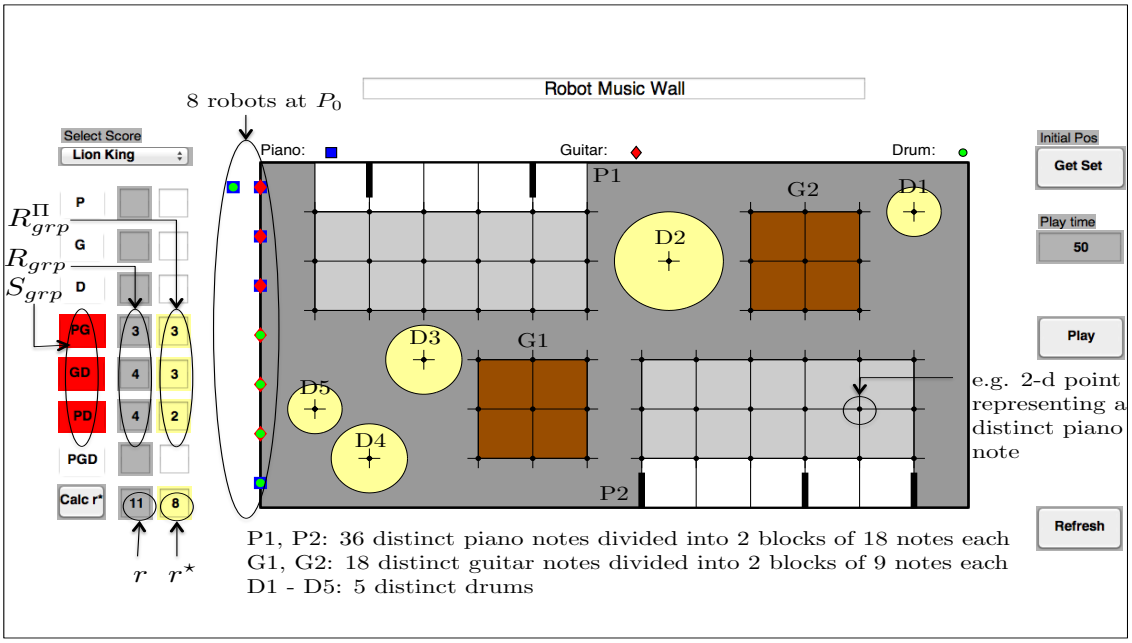


Figure 13: A user-interface developed for the Robot Music Wall 2.0, with coordinates representing either piano notes, guitar notes or drums. For a user specified choice of $S_{grp} = \{\{p, g\}, \{g, d\}, \{p, d\}\}$ and $R_{grp} = \{3, 4, 4\}$, *MinBots* provides $r^\star = 8$, and $R^{\Pi}_{grp} = \{3, 3, 2\}$, while *PathGen* provides the path of every robot corresponding to R^{Π}_{grp}. Each robot is color indexed to denote the instruments it can play, and is initially positioned at the boundary of the wall.

We created the heterogeneous Score associated with the Lion King song, and for a user specified set of resources, i.e. (S_{grp}, R_{grp}) (enumerating the *available* skill sets, and the *number* of robots per skill set), we calculated the minimum number of robots, $r^\star$, required to execute the Score. Additionally, we calculated a corresponding distribution of robots per skill set, R^{Π}_{grp} (need not be unique), using the *MinBots* algorithm, and their trajectories, using the *PathGen* algorithm. Similar to previous cases, the instant a robot reached an assigned timed position, it was encircled by a light circle (yellow), and the sound of the corresponding instrument was generated. Thus, we created a GUI for calculating $r^\star$ as a function of user specified available resources, and demonstrated the routing problem

through a performance of “Can You Feel the Love Tonight” on the Robot Music Wall 2.0. Instances of one such simulation are shown in Figure 14.

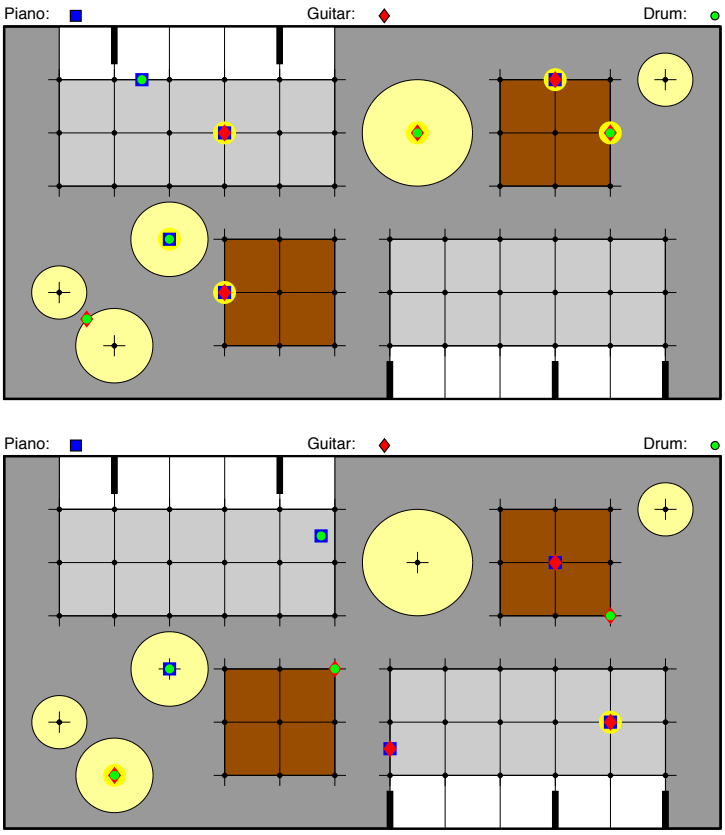


Figure 14: Two snapshots of the execution of the song “Can You Feel the Love Tonight”, being played by eight robots.

CHAPTER V

A DISTRIBUTED FRAMEWORK

A frequent requirement in many multi-robot applications, is the need for a distributed framework, since an infrastructure that supports a centralized authority for large teams of mobile robots is often, not a feasible option. It is preferable that robots coordinate with one another to allocate and execute individual tasks, through an efficient, distributed mechanism. Thus, in an effort to extend the scope of our work to further include such multi-robot scenarios, we consider spatio-temporal routing in the context of a distributed framework [20].

Since the theory of assignment problems underlies most of our work up until this point, we analyze the basic Linear Sum Assignment Problem (LSAP) [55] in a distributed setting. In the light of spatio-temporal multi-robot routing, we perceive the LSAP as simply a "one-step" assignment problem between robot positions, and timed positions specified at some time instant t_i in the Score. As a natural extension of our work, in which we frequently use the Hungarian Method [44], we develop a novel algorithm - essentially a distributed version of the Hungarian Method, for analyzing the LSAP in a distributed setting. Furthermore, we extend the distributed algorithm towards dynamic spatio-temporal routing, where the Score changes in real-time and the robots adapt their routes accordingly.

5.1 Distributed Assignment

In this section, we consider the Linear Sum Assignment Problem (LSAP) in a distributed environment, where the LSAP comprises of finding a one-to-one matching between robots and spatial locations (targets), while optimizing the total distance traveled. Using a weak

connectivity assumption, where robots communicate locally with adjacent robots via a dynamic connected directed information exchange network (that can be generalized to jointly connected networks as well), we propose a distributed version of the Hungarian Method, and show that all robots converge to identical[1] optimal solutions, without any coordinator or shared memory. Moreover, we provide precise results on the number of iterations required for convergence, and the amount of information communicated.

In the following section, we review the LSAP under a centralized setting, before we delve into its proposed distributed counterpart.

5.1.1 A Review of the Assignment Problem and the Hungarian Method

We review some key definitions and theorems, used to express the general form of the LSAP in graph theoretic terms, and to understand the Hungarian Method employed for solving it.

- **Bipartite Graph:** A graph $G = (V, E)$, where the vertex set V is decomposed into two disjoint sets of vertices A and B respectively, such that no two vertices in the same set are adjacent. In general, we say that the graph G has bipartition (A, B).

- **Matching:** A set of edges without common vertices.
- **Maximum Cardinality Matching:** A matching that contains the largest possible number of edges.
- **Vertex Cover:** A set of vertices such that each edge is incident on at least one vertex of the set.
- **Minimum Vertex Cover:** A vertex cover that contains the smallest possible number of vertices.

Remark 1 *In a bipartite graph, the number of edges in a maximum cardinality matching equals the number of vertices in a minimum vertex cover (by Konig's theorem [42]). In*

[1]The assignment problem may have multiple optimal solutions

fact, due to this inter-relation between a matching and a vertex cover, algorithms used for finding a maximum cardinality matching M (e.g. Hopcroft-Karp [35]), can be extended to finding a corresponding minimum vertex cover $V_c \subset V$.

Using the definitions presented above, we review the formal, graph theoretic interpretation of the LSAP, known as the **Minimum Weight Bipartite Matching Problem** [55]:

"Given a graph $G = (V, E)$ with bipartition (A, B) and weight function $w : E \to \mathbb{R}$, the objective is to find a maximum cardinality matching M of minimum cost, where the cost of matching M is given by $c(M) = \sum_{e \in M} w(e)$".

Without loss of generality, we can assume that G is *complete*[2], i.e. there exists an edge between every vertex $a \in A$, and every vertex $b \in B$, and *balanced*[3], i.e. $|A| = |B| = |V|/2$. Hence, a maximum cardinality matching M is always a perfect matching, i.e. $|M| = |V|/2$. Next, we review the **dual** of the Minimum Weight Bipartite Matching Problem, stated as follows:

"Given a graph $G = (V, E)$ with bipartition (A, B), a weight function $w : E \to \mathbb{R}$, and a vertex labeling function $y : V \to \mathcal{R}$, the objective is to find a feasible labeling of maximum cost, where a feasible labeling is a choice of labels y, such that $w(a, b) \geq y(a) + y(b) \, \forall \, (a, b) \in E$, and the cost of the labeling is given by $c(y) = \sum_{a \in A} y(a) + \sum_{b \in B} y(b)$".

Moreover, given a feasible labeling y, an **equality subgraph** $G_y = (V, E_y)$ is a subgraph of G such that,

$$E_y = \{(a, b) \,|\, y(a) + y(b) = w(a, b)\} \tag{68}$$

Now that we have discussed the Minimum Weight Bipartite Matching Problem, as well as its corresponding dual, we review a key theorem that provides the basis for the Hungarian Method, the first primal-dual algorithm developed for solving the LSAP. We follow the

[2] by adding high-weight edges, as per the big-M method [12, 55], if required.
[3] by adding dummy vertices with associated edge weights 0, if required.

theorem with a brief description of the Hungarian Method that will assist us in explaining our proposed distributed algorithm in later sections of this chapter.

Theorem 5 (Kuhn-Munkres) *Given a bipartite graph $G = (V, E)$ with bipartition (A, B), a weight function $w : E \to \mathbb{R}$, and a vertex labeling function $y : V \to \mathbb{R}$, let M and y be feasible (M is a perfect matching and y is a feasible labeling). Then M and y are optimal if and only if $M \subseteq E_y$, i.e. each edge in M is also in the set of equality subgraph edges E_y, given by (68).*

Initialization: Given a graph $G = (V, E)$ with bipartition (A, B), and a weight function w (Figure 29a), the Hungarian Method begins with an arbitrary feasible labeling y (Figure 29b), generates the corresponding equality subgraph edges E_y using (68) (Figure 15c), finds a maximum cardinality matching $M_y \subseteq E_y$, and a corresponding minimum vertex cover $V_{c_y} \subset V$, with bipartition (A_{c_y}, B_{c_y}) as per Remark 1 (Figure 15d). The algorithm then performs the following two-step iterations repeatedly, until M_y is a perfect matching:

1. - The algorithm uses V_{c_y} to isolate a set of candidate edges $E_{cand} \subseteq E \setminus E_y$ as per Remark 2, and calculates the following (Figure 16a):

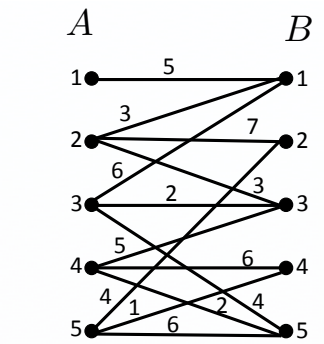
$$\delta = min_{(a,b) \in E_{cand}} slack(w, y, a, b)$$

where $slack(w, y, a, b) = w((a, b)) - (y(a) + y(b))$

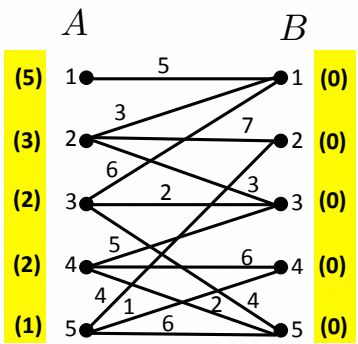
- Using δ and V_{c_y}, the algorithm updates y as follows (Figure 16b):

$$y(a) = y(a) - \delta, \ \forall a \in A_{c_y}$$
$$y(b) = y(b) + \delta, \ \forall b \in B \setminus B_{c_y}$$

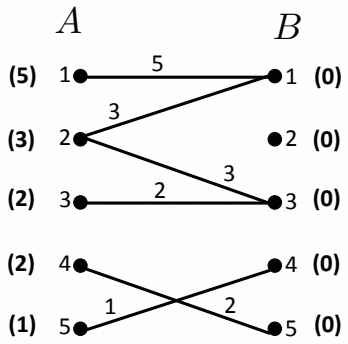
2. For the updated y, the algorithm finds the corresponding equality subgraph edges E_y using (68) (Figure 16c), to find a maximum cardinality matching $M_y \subseteq E_y$, and a corresponding minimum vertex cover $V_{c_y} \subset V$ as per Remark 1 (Figure 16d).



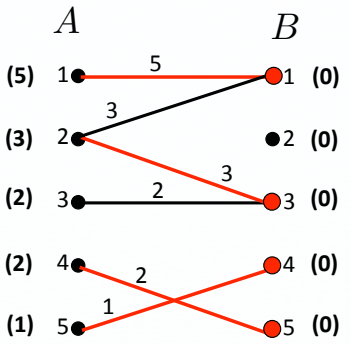
(a) An example of a bipartite graph $G = (V, E)$, with edge weight function $w : E \to \mathbb{R}$.



(b) Initial feasible vertex labeling function $y : V \to \mathbb{R}$ (highlighted in yellow).

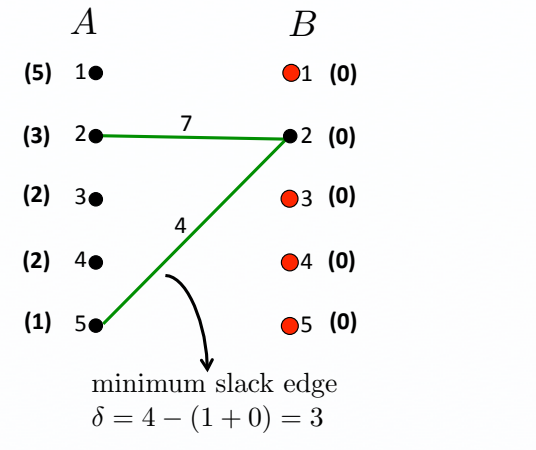


(c) For the given y, the corresponding set of equality subgraph edges E_y.

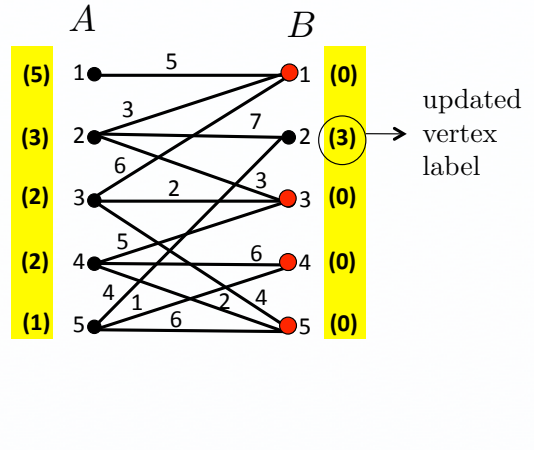


(d) For the given E_y, a maximum cardinality matching M_y (red edges), and a corresponding minimum vertex cover $V_{c_y} = (A_{c_y}, B_{c_y})$ (red vertices).

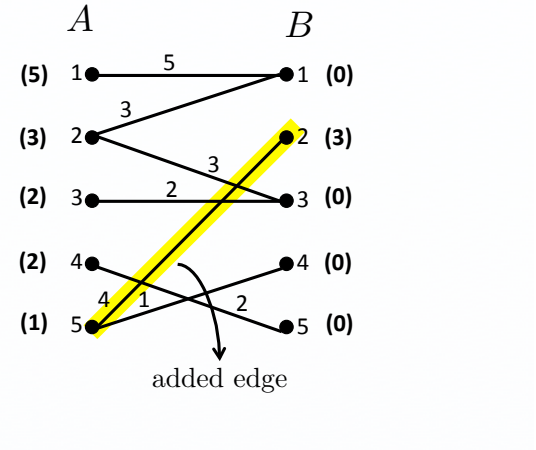
Figure 15: An example of the Initialization step of the Hungarian Method.



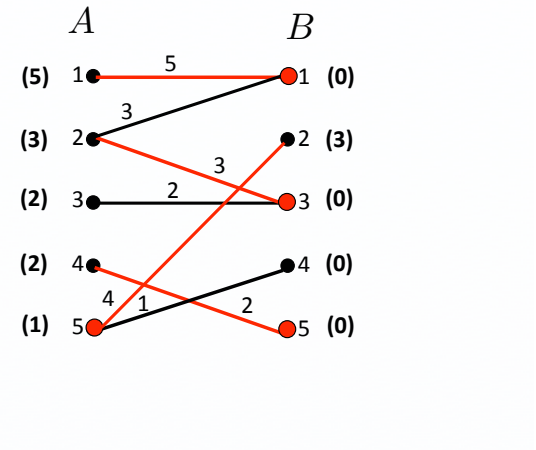
(a) Step 1: For the given minimum vertex cover V_{c_y}, the isolated set of candidate edges (green edges). Using such edges, the minimum slack δ is calculated.



(b) Step 1: The updated feasible vertex labeling function y (highlighted in yellow), using the minimum slack δ.



(c) Step 2: For the updated y, the corresponding set of equality subgraph edges E_y (with the new, added edge highlighted in yellow).



(d) Step 2: For the updated E_y, a maximum cardinality matching M_y (red edges), and a corresponding minimum vertex cover $V_{c_y} = (A_{c_y}, B_{c_y})$ (red vertices).

Figure 16: An example of a two-step iteration of the Hungarian Method.

Remark 2 *As mentioned above in Step 1, the selection of the candidate edges is done based on the minimum vertex cover $V_{c_y} = (A_{c_y}, B_{c_y})$. In particular, the set of candidate edges E_{cand} represents the edges between vertices in $A \setminus A_{c_y}$ and vertices in $B \setminus B_{c_y}$, i.e. edges between uncovered vertices in A and uncovered vertices in B (see Figure 16a for an example).*

Without delving into details, we provide a quick proof sketch that shows the Hungarian Method converges to an optimal solution, for reference purposes in later sections of this chapter (see [55] for details).

Lemma 4 *Given $G = (V, E)$ with bipartition (A, B), a weight function w, a feasible vertex labeling function y, and a corresponding maximal matching M_y, every two-step iteration (steps (a) and (b)) of the Hungarian Method results in the following: (i) An updated y that remains feasible. (ii) An increase in the matching size $|M_y|$, or no change in the matching M_y, but an increase in $|A_{c_y}|$ (and corresponding decrease in $|B_{c_y}|$, such that $|A_{c_y}| + |B_{c_y}| = |M_y|$).*[4]

The above stated Lemma 4 ensures that the size of a matching M_y increases after a finite number of two-step iterations (worst-case $|V|^2$). Since the algorithm converges when M_y is perfect, i.e. $|M_y| = |R|$, Lemma 4 in conjunction with Theorem 5 proves that the Hungarian Method converges to an optimal solution (perfect matching with minimum cost), after $\mathcal{O}(|V|^2)$ two-step iterations. Each two-step iteration requires $\mathcal{O}(|V|^2)$ time, yielding a total running time of $\mathcal{O}(|V|^4)$ (through certain modifications, this running time can be reduced to $\mathcal{O}(|V|^3)$). Now that we have reviewed the LSAP, as well as the Hungarian Method used for solving it, we proceed to setup the distributed assignment problem central to this chapter.

[4] *Either $|A_{c_y}|$ increases and $|B_{c_y}|$ decreases, or $|B_{c_y}|$ increases and $|A_{c_y}|$ decreases, depending on the particular implementation of the algorithm employed for finding M_y and V_{c_y}.*

5.1.2 Distributed Problem Setup

Let $R = \{1, 2, ..., r\}$ denote a set of r robots, and $P = \{1, 2, ..., p\}$ denote a set of p targets. Let $P^i \subseteq P$ be the set of targets that robot $i \in R$ can be assigned to, with the associated cost function $c^i : P^i \to \mathbb{R}$. Note that $c^i(j \in P^i)$ represents the distance between robot i's location, and target j's location. Every target in P can be assigned to at least one robot, i.e. $\cup_{i \in R} P^i = P$.

We assume that each robot $i \in R$ knows the sets R and P. Moreover, each robot knows the cost function c^i, associated with the subset of targets that it can be assigned to (see Figure 18a for an example).

Communication network: We model the communication between the robots by a time-varying directed graph $\mathcal{G}_c(t) = (R, E_c(t))$, where $t \in \mathbb{R}_{\geq 0}$. In such a graph, an edge exists from robot i to robot j at some time t *if and only if* robot i transfers information to robot j at time t. Moreover, for robot i, we let $\mathcal{N}_O(i, t)$ denote the set of outgoing neighbors, and $\mathcal{N}_I(i, t)$ denote the set of the incoming neighbors respectively. Based on the above discussion, we assume the following:

Assumption 3 *For every time instant $t \in \mathbb{R}_{\geq 0}$, the directed graph $\mathcal{G}_c(t)$ is strongly connected (there exists a directed path from every robot, to every other robot in $\mathcal{G}_c(t)$)* [5].

We are interested in the problem of assigning robots to targets with minimum total cost, where each robot $i \in R$ initially knows (R, P, c^i), and can communicate with other robots *only* via the time-varying communication graph $\mathcal{G}_c(t)$, as per Assumption 3. Note that we can generate the problem data for the corresponding centralized assignment problem (Section 5.1.1) as follows:

- Bipartite graph $G = (V, E)$ with bipartition (R, P), where $E = \{(i, j) \mid j \in P^i\}\ \forall i \in R$

[5] *One can extend Assumption 3 to the case where $\mathcal{G}_c(t)$ is jointly connected over some time period, w.l.o.g.*

- Weight function $w : E \to \mathbb{R}$, where $w((i,j)) = c^i(j), \forall (i,j) \in E$

As mentioned before, we can modify G to ensure its balanced and complete. For now, assume $|R| = |P| = |V|/2$, and include high-weight edges as per the big-M method, to make G complete. Recall that the optimal solution to such an assignment problem is a minimum weight perfect matching. However, due to the inherent degeneracy in assignment problems, there can be multiple minimum weight perfect matchings. Let $\mathcal{M}$ denote the set of such minimum weight perfect matchings. Then, for any $M \in \mathcal{M}$, the corresponding *unique* optimal cost $c^\star$ is given by $c^\star = c(M) = \sum_{e \in M} w(e)$.

Note that the property of degeneracy in assignment problems can be of particular concern, especially in context to a distributed framework, since all robots must converge not only to an optimal solution (assignment), but to the *same* optimal solution. We denote such a solution by $\hat{M} \in \mathcal{M}$ (note that $c(\hat{M}) = c^\star$). Thus, we define the distributed version of the assignment problem as follows:

Distributed Assignment Problem: Given a set of robots R, a set of targets P, and a communication graph $\mathcal{G}_c(t)$ as per Assumption 3. Every robot $i \in R$ knows (R, P, c^i), i.e. the sets R and P, and the cost function associated with itself and targets that it can be assigned to. Then, the distributed assignment problem requires all robots to converge to identical optimal solutions, $\hat{M} \in \mathcal{M}$, that are also optimal to the corresponding centralized assignment problem.

5.1.3 A Distributed Version of the Hungarian Method

Drawing from the description of the Hungarian Method discussed previously in Section 5.1.1, we propose the *Distributed-Hungarian* algorithm for solving the Distributed Assignment Problem (stated above). Though we provide a formal proof of convergence later in this chapter, for the sake of clarity, we emphasize the gist of our algorithm as follows:

Similar to the Hungarian Method in the centralized case, the idea behind the *Distributed-Hungarian* algorithm is to execute a finite number of unique stages or two-step iterations (1 and 2), required to converge to an optimal solution. However, the execution of these stages is distributed across the robots. In particular, each robot shares and operates on certain limited information as opposed to complete information in the centralized case, executing a stage (two-step iteration) as and when it has enough information to do so. Moreover, by sharing its updated (evolved) information across the network, another robot is able to execute a subsequent stage, as and when it has enough information, and so on and so forth. Every stage is executed at least once, by at least one robot. In the following paragraphs, we provide a more detailed description of the *Distributed-Hungarian* algorithm.

All robots begin running their individual copies of the algorithm at some initial time $t_0 \in \mathbb{R}_{\geq 0}$, and continue to run it synchronously, at intervals of T_s seconds ($T_s \in \mathbb{R}_{>0}$). The robots maintain and update a corresponding global iteration-counter $\alpha \in \mathbb{N}$ (initialized at 1), that represents the α^{th} iteration of the algorithm, executed synchronously at time $t_0 + (\alpha - 1)T_s$.

Each robot $i \in R$ starts with the initial information (R, P, c^i), and creates a bipartite graph $G^i_{orig} = (V, E^i_{orig})$ with bipartition (R, P), and weight function $w^i_{orig} : E_{orig} \to \mathbb{R}$ (see the *Initialize* sub-algorithm for details, and Figures 18b-18c for an example). Moreover, throughout the execution of the algorithm, robot i shares, updates and operates on a bipartite graph $G^i_{lean} = (V, E^i_{lean})$ with bipartition (R, P), a corresponding weight function $w^i_{lean} : E^i_{lean} \to \mathbb{R}$, and a corresponding vertex labeling function $y^i : V \to \mathbb{R}$ (the subscript *lean* refers to the sparseness of the graph, with significantly less number of edges than $|V|^2$, as in the centralized case).

In particular, the graph G^i_{lean} contains sparse representations of two kinds of edges, one corresponding to the equality subgraph edges E^i_y (necessary for Step 2 of the Hungarian Method), and the other corresponding to candidate edges E^i_{cand} (necessary for Step 1 of the

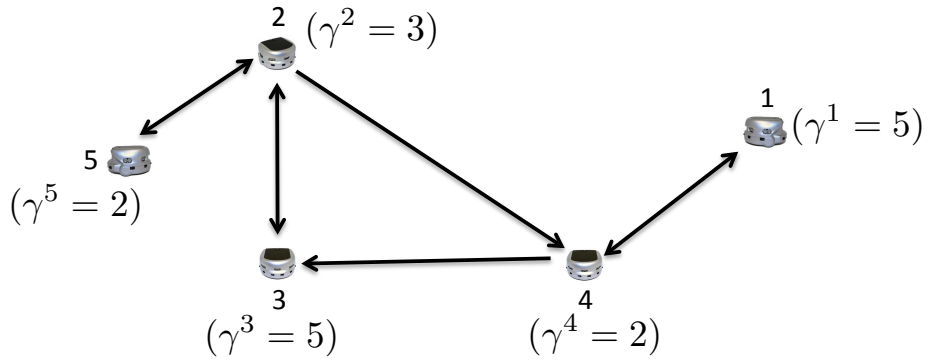


Figure 17: An example of 5 robots, communicating via a strongly connected directed graph on some iteration α of the *Distributed-Hungarian* algorithm, with corresponding progress-counter values.

Hungarian Method). We denote such disjoint sets of edges by $E_{lean} = (E_y, E_{cand})$. Moreover, robot i maintains and updates a so called progress-counter, γ^i, denoting the cumulative number of *completed* stages (two-step iterations) of the Hungarian Method, resulting in its current information G^i_{lean}, w^i_{lean} and y^i. In other words, γ^i measures (informally speaking) robot i's "progress" towards finding a solution.

On every iteration α of the *Distributed-Hungarian* algorithm, the following two actions are performed by all robots:

- (**Send**) Robot i sends a message msg^i, to each of its outgoing neighbors, where $msg^i = (G^i_{lean}, w^i_{lean}, y^i, \gamma^i)$ (see Figure 18c for an example of an initial message sent by a robot).

- (**Receive and Compute**) Robot i receives the messages from its incoming neighbors, and through computations involving the received information, its own message msg^i, and its original information (G^i_{orig}, w^i_{orig}), robot i prepares its new message for sending.

As mentioned earlier, a key feature of the *Distributed-Hungarian* algorithm is that different stages of the Hungarian Method are performed across different robots. Depending

on the underlying communication graph (as per Assumption 3), and consequently the information that different robots have access to, the robots can progress at different rates from each other. Thus, on some iteration α of the *Distributed-Hungarian* algorithm, robots may have different counter values (see Figure 17).

Every robot extracts specific information from the messages it receives from its incoming neighbors. In particular, robot i only uses information from robots that have the highest counter value among all received counters (including its own), since their information is most evolved in terms of the number of *completed* stages (two-step iterations) of the Hungarian Method. Robot i combines such information (see the *Parse-Info* sub-algorithm for details) into an updated (G^i_{lean}, w^i_{lean}, y^i, γ^i), and *continues* with the Hungarian Method (performs a subsequent two-step iteration if possible, where Steps 1 and 2 are described in detail, in the *Update-Label*, and the *Find-Matching-Cover* sub-algorithms respectively). The caveat is that robot i must have sufficient candidate edges in E^i_{cand}, for it to be able to execute Step 2. If this is not the case, robot i gathers whatever pertinent candidate edges it has (drawing an edge if required, from its original information (G^i_{orig}, w^i_{orig})), and updates E^i_{cand} accordingly (explained in detail, in the *Get-Ecand* sub-algorithms).

Robot i then prepares its new message for sending, at the next synchronized iteration of the *Distributed-Hungarian* algorithm. This process is repeated until robot i finds a perfect matching M^i_y on executing Step 2, at which point it repeatedly sends the corresponding message till some stopping criterion (presented later in this chapter) is met. Figures 19 - 20 depict an example of an instance of the *Distributed-Hungarian* algorithm, where a robot does not have enough candidate edges to perform a two-step iteration of the Hungarian Method, while Figures 21 - 23 depict an example where a robot can, in fact, perform a two-step iteration of the Hungarian Method.

Remark 3 *The Distributed-Hungarian algorithm requires every robot i to repeatedly share as well as update a sparse graph $G^i_{lean} = (V, E^i_{lean})$, with the edge weight function w^i_{lean},*

and the vertex labeling function y^i. The structure of this graph is very important, as it is contingent to many results presented later in this chapter, regarding (i) the amount of information transmitted, (ii) robot i's convergence to an optimal solution, and (iii) elimination of degeneracy in the final solution across all robots. Thus, it is important to note that the edges in G^i_{lean}, i.e. sparse representations of equality subgraph edges and candidate edges, are initialized in the Initialize sub-algorithm, and updated using the Find-Matching-Cover sub-algorithm, and the Get-Ecand sub-algorithm respectively.

Algorithm *Distributed-Hungarian* $(R, P, c^i, \mathcal{G}_c)$

Inputs:
Set of robots R, set of targets P
Robot i's cost function $c^i : P^i \to \mathbb{R}$
Time-varying communication graph $\mathcal{G}_c$ (Assumption 3.)

$(msg^i, G^i_{orig}, w^i_{orig}) \leftarrow$ ***Initialize*** (R, P, c^i) {see Figure 18}

$\alpha \leftarrow 1$ {Initialize iteration-counter α}

while $\neg$ stopping criterion **do** {See Corollary 3.}

(Send)
Robot i sends msg^i to every outgoing neighbor $k \in \mathcal{N}_O(i, t_0 + (\alpha - 1)T_s)$, every T_s seconds

(Receive and Compute)
$\mathcal{I} \leftarrow \bigcup_{j \in \mathcal{N}_I(i, t_0+(\alpha-1)T_s)} \{msg^j\} \cup msg^i$
{$\mathcal{I}$ contains messages received by incoming neighbors of robot i at time $t_0 + (\alpha - 1)T_s$}

$(G^i_{lean}, w^i_{lean}, y^i, \gamma^i) \leftarrow$ ***Parse-Info*** $(\mathcal{I})$
{$G^i_{lean} = (V, E^i_{lean})$, where $V = (R, P)$, and $E^i_{lean} = (E^i_y, E^i_{cand})$}

If $\gamma = -1$, go to line 21 {Not ready yet, to enter the computation phase}

(**Begin Computation**)

$(V^i_{c_y}, M^i, E^i_y) \leftarrow$ ***Find-Matching-Cover*** (V, E^i_y) {For the input graph $G = (V, E^i_y)$, $V^i_{c_y} = (R^i_{c_y}, P^i_{c_y})$ is a minimum vertex cover, and $M^i \subseteq E^i_y$ is a maximal matching (see Figures 19d and 21d)}

if M^i is not perfect **then**

$E^i_{cand} \leftarrow$ ***Get-Ecand*** $(G^i_{orig}, w^i_{orig}, y^i, V^i_{c_y}, E^i_{cand})$ {see Figures 20a - 20c, and 22a -22c}

if $\exists j \in P \setminus P^i_{c_y}$ such that $(i, j) \in E^i_{cand}$, $\forall i \in R \setminus R^i_{c_y}$ **then** {From every uncovered robot, there is an edge in E^i_{cand}, to an uncovered target (see Figure 22c)}

$(y^i, E^i_y) \leftarrow$ ***Update-Label*** $(G^i_{lean}, w^i_{lean}, y^i, V^i_{c_y}, E^i_{cand})$ {**Step** 1, e.g. Figures 22d and 23a}

$(V^i_{c_y}, M^i, E^i_y) \leftarrow$ ***Find-Matching-Cover*** (V, E^i_y) {**Step** 2, e.g. Figure 23b}

$\gamma^i \leftarrow \gamma^i + 1$ {Update the progress-counter}

$E^i_{cand} \leftarrow$ ***Get-Ecand*** $(G^i_{orig}, w^i_{orig}, y, V_{c_y}, \emptyset)$ {see Figure 23c}

end if

end if

For the (possibly) updated set of edges $E^i_{lean} = (E^i_y, E^i_{cand})$, update the corresponding weight function $w^i_{lean} : E^i_{lean} \to \mathbb{R}$

$msg^i \leftarrow (G^i_{lean}, w^i_{lean}, y^i, \gamma^i)$ {see Figures 20d and 23d}

$\alpha \leftarrow \alpha + 1$ {End of iteration α; Increment the iteration-counter}

end while

In subsequent paragraphs, we provide an overview of each sub-algorithm used in the *Distributed-Hungarian* algorithm.

Initialize **(R, P, c^i):** Given robot i's initial information (R, P, c^i), the *Initialize* sub-algorithm generates a bipartite graph G^i_{orig} with weight function w^i_{orig}, and an initial sparse graph G^i_{lean}, with weight function w^i_{lean} and vertex labeling function y^i (see Figure 18 for an example). Moreover, the sub-algorithm sets robot i's progress-counter to -1.

Sub-algorithm *Initialize* (R, P, c^i)

1: Inputs:
Set of robots R, set of targets P
Robot i's cost function $c^i : P^i \to \mathbb{R}$

2: $G^i_{orig} \leftarrow (V, E^i_{orig})$, where $E^i_{orig} \leftarrow \{(i, j) \mid j \in P\}$ {Bipartite graph, with bipartition (R, P)}

3: $w^i_{orig}((i, j) \in E^i_{orig}) \leftarrow \begin{cases} c^i(j)\ \forall j \in P^i \\ \mathfrak{M}\ \forall j \in P \setminus P^i \end{cases}$
{Weight function $w^i_{orig} : E^i_{orig} \to \mathbb{R}$, with $\mathfrak{M}$ denoting the cost to targets that cannot be assigned to robot i}

4: $y^i(v \in V) \leftarrow 0, \forall v \in V$ {Vertex labeling function $y^i : V \to \mathbb{R}$, initialized to 0}

5: $E^i_y \leftarrow \{(i, j^\star)\},\ j^\star \in P$, s. t. $w^i_{orig}(i, j^\star) = \min_{j \in P} w^i_{orig}(i, j)$ {E^i_y represents the edge between robot i and the target with minimum cost associated with robot i (We denote the target by $j^\star$)}

6: $E^i_{cand} \leftarrow \emptyset$

7: $G^i_{lean} \leftarrow (V, E^i_{lean})$, where $E^i_{lean} \leftarrow (E^i_y, E^i_{cand})$

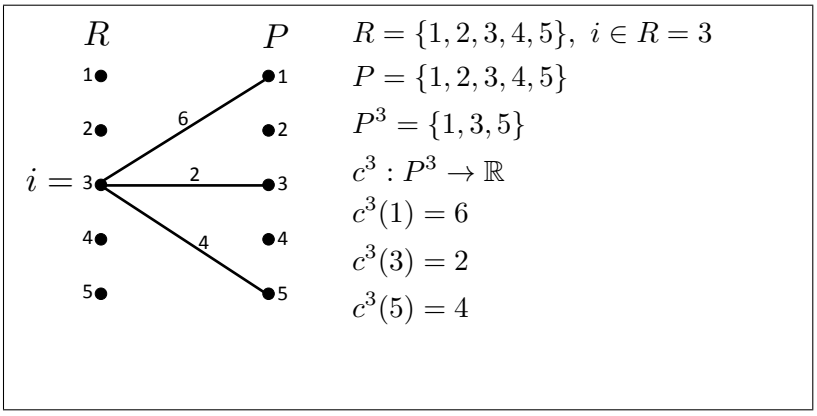
8: $w^i_{lean} \leftarrow w^i_{orig}|_{E^i_{lean} \subseteq E^i_{orig}}$

9: $\gamma^i \leftarrow -1$ {Initialize progress-counter}

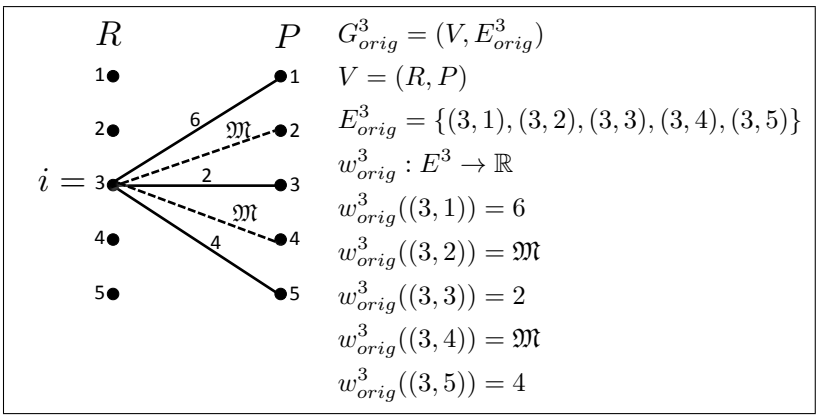
10: $msg^i \leftarrow (G^i_{lean}, w^i_{lean}, y^i, \gamma^i)$

11: **return** $msg^i, G^i_{orig}, w^i_{orig}$

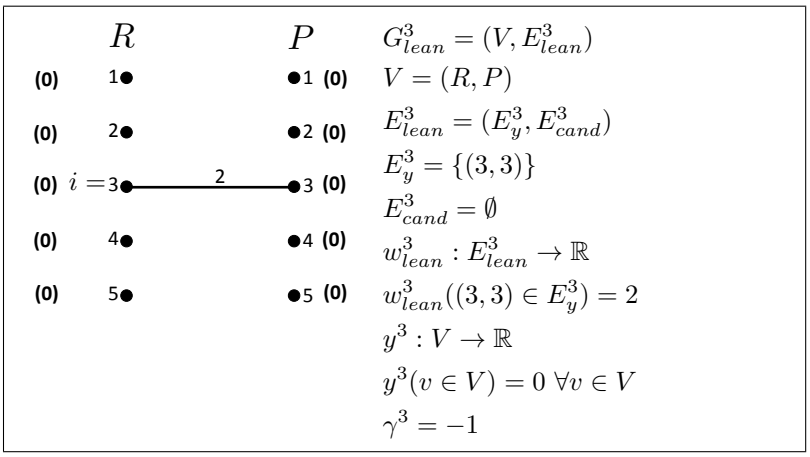
Parse-Info **($\mathcal{I}$):** Given $\mathcal{I}$, containing messages from a subset of robots, say $R' \subseteq R$, where a robot i's message is the following: $msg^i = (G^i_{lean}, w^i_{lean}, y^i, \gamma^i)$, the *Parse-Info* sub-algorithm parses and combines the messages into a resultant sparse graph $G_{lean} = (V, E_{lean})$, with weight function w_{lean} and vertex labeling y, and a progress-counter value γ. The gist of this sub-algorithm is to utilize the information of *only* those robots that



(a) Robot 3's original information, (R, P, c^3).



(b) Robot 3's original information in the form of the graph G^3_{orig}, and edge weight function w^3_{orig}.



(c) Robot 3's initial message, $msg^3 = (G^3_{lean}, w^3_{lean}, y^3, \gamma^3)$.

Figure 18: For the example in Figure 29a, robot 3's initial information, and the corresponding output of the *Initialize* sub-algorithm, comprising of the bipartite graph G^3_{orig}, the corresponding weight function w^3_{orig} and the outgoing message msg^3.

have the maximum progress-counter value amongst all counter values. In particular the following two cases can occur:

1. *The maximum progress-counter value amongst all received counter values is* -1*:* Such a case implies that every robot's message in $\mathcal{I}$ is its *initial* message (generated by the *Initialize* sub-algorithm). In this case, the *Parse-Info* sub-algorithm combines the edges E_y^i, of every robot $i \in R'$, into a resultant graph $G_{lean} = (V, E_{lean})$, with corresponding weight function w_{lean}, where $E_{lean} = (E_y, E_{cand})$, and $E_y = \bigcup_{i \in R'} E_y^i$.

   - *Special Case:* If E_y contains an edge from *every* robot $i \in R$, then the *Parse-Info* sub-algorithm computes an initial vertex labeling function y as per line 6, and updates the progress-counter value from -1 to 0, denoting that the information $(G_{lean}, w_{lean}, y, \gamma)$ is ready for computation (line 9 of the *Distributed-Hungarian* algorithm).

2. *The maximum progress-counter value amongst all received counter values is some value* $k \in \mathbb{N}_0$*:* In such a case, the *Parse-Info* sub-algorithm isolates a set of robots $R_{lead} \subseteq R'$, with corresponding progress-counter values equal to k, and for any one robot $i \in R_{lead}$, the sub-algorithm sets $y = y^i$ and $E_y = E_y^i$. (We prove later in this chapter, that if two robots have identical counter values, then they have identical vertex labeling functions and equality subgraph edges, and hence it does not matter which robot is chosen to set y and E_y). However, the sub-algorithm combines the candidate edges of *every* robot $i \in R_{lead}$, i.e. $E_{cand} = \bigcup_{i \in R_{lead}} E_{cand}^i$, and updates the weight function w_{lean}, for the corresponding edge set $E_{lean} = (E_y, E_{cand})$. Moreover, the sub-algorithm updates the counter value γ to k.

Sub-algorithm *Parse-Info* ($\mathcal{I}$)

1: **Inputs**:
$\mathcal{I}$ containing messages from a subset of robots, say $R' \subseteq R$, where a robot i's message is the following: $msg^i = (G^i_{lean}, w^i_{lean}, y^i, \gamma^i)$

2: $\gamma \leftarrow \max_{i \in R'} \gamma^i$ {progress-counter with the highest value}

3: **if** $\gamma = -1$ **then**

4: $E_y \leftarrow \bigcup_{i \in R'} E^i_y$ {Collect initial edges from every robot in R'}

5: **if** $|E_y| = |R|$ **then** {The resulting (combined) set of edges E_y contains the edge E^i_y from *every* robots initial outgoing message msg^i.}

6: Generate the initial vertex labeling function $y : V \rightarrow \mathbb{R}$ as follows:
$y(i) \leftarrow w^k_{lean}((i,j) \in E^k_y), \forall i \in R$, where robot $k \in R'$ is a robot that contains the edge (i,j) in its set of sent edges

$y(j) \leftarrow 0, \forall j \in P$

7: $\gamma \leftarrow 0$ {Set the progress-counter γ to 0, signifying that the received information is *complete* and ready for the computation phase}

8: **end if**

9: **else**

10: $R_{lead} \leftarrow \{i \,|\, \gamma^i = \gamma\}$ {Subset of robots with counters corresponding to the highest value}

11: Choose a robot $i \in R_{lead}$,
$y \leftarrow y^i$, $E_y \leftarrow E^i_y$ {All robots in R_{lead} have identical vertex labeling functions and equality subgraph edges (proved later in this chapter)}

12: **end if**

13: $E_{cand} \leftarrow \bigcup_{i \in R_{lead}} E^i_{cand}$ {Collect candidate edges from every robot in R_{lead}}

14: $G_{lean} \leftarrow (V, E_{lean})$, where $E_{lean} \leftarrow (E_y, E_{cand})$

15: $w_{lean}((i,j) \in E_{lean}) \leftarrow w^k_{lean}((i,j) \in E^k_{lean}), \forall (i,j) \in E_{lean}$, where robot $k \in R_{lead}$ is a robot that contains the edge (i,j) in its set of sent edges

16: **return** $G_{lean}, w_{lean}, y, \gamma$

Find-Matching-Cover **(V, E_y):** Given a bipartite graph $G_y = (V, E_y)$, with bipartition (R, P), the *Find-Matching-Cover* sub-algorithm finds a maximum cardinality matching M_y, and a corresponding minimum vertex cover $V_{c_y} = (R_{c_y}, P_{c_y})$, as per the Initialization and Step 2 of the Hungarian Method (see Figures 19d, 21d, and 23b for examples).

As mentioned in Remark 3, an important aspect of the *Find-Matching-Cover* sub-algorithm is that it returns a sparse representation of the inputted edges E_y, denoted by $E'_y \subseteq E_y$. In particular, E'_y satisfies the following:

1. $|E'_y| = |R_{c_y}| + |V|/2$
2. For the bipartite graph $G' = (V, E'_y)$, the *Find-Matching-Cover* sub-algorithm returns a maximum matching M'_y, and minimum vertex cover V'_{c_y}, identical to the original matching and cover, i.e. $M'_y = M_y$, and $V'_{c_y} = V_{c_y}$.

Sub-algorithm *Find-Matching-Cover* (V, E_y)

Inputs:
Bipartite graph $G_y = (V, E_y)$, with bipartition (R, P), where for each robot $i \in R$, there exists at least one target $j \in P$ such that $(i, j) \in E_y$
$M \leftarrow$maximal matching given G_y {Using e.g. Hopcroft-Karp [35]}
$E_t, E'_y \leftarrow \emptyset$
for every unmatched target $j \in P$ **do**
    $E_t \leftarrow E_t \cup \{$edges of the alternating tree with root $j\}$.
end for
$R_{c_y} \leftarrow \{i \in R \,|\, (i, j) \in E_t\}$
$P_{c_y} \leftarrow \{j \in P \,|\, (i, j) \in M \setminus E_t\}$
for every $i \in R_{c_y}$ **do** {From every robot, choose an edge in E_t that is not in M}
    Choose some target $j \in P$ such that $(i, j) \in E_t \setminus M$
    $E'_y \leftarrow E'_y \cup \{(i, j)\}$
end for
$E'_y \leftarrow E'_y \cup M$
for every unmatched robot $i \in R$ **do**
    Choose some target $j \in P$ such that $(i, j) \in E_y$
    $E'_y \leftarrow E'_y \cup \{(i, j)\}$
end for
return $V_{c_y} \leftarrow (R_{c_y}, P_{c_y}), M, E'_y$

Get-Ecand **($G^i_{orig}, w^i_{orig}, y, V_{c_y}, E_{cand}$):** Given robot i's original information, in the form of a bipartite graph $G^i_{orig} = (V, E^i_{orig})$, with bipartition (R, P), a corresponding weight function w^i_{orig}, a vertex labeling function y and a corresponding minimum vertex cover $V_{c_y} = (R_{c_y}, P_{c_y})$, the *Get-Ecand* sub-algorithm isolates a set of candidate edges, as per Remark 1, and chooses *exactly one edge* with minimum slack (see line 4). Such an edge is robot i's contribution to the sparse set of inputted candidate edges E_{cand} (see Figures 20a - 20c, 22a -22c, and 23c for examples).

Sub-algorithm *Get-Ecand* $(G^i_{orig}, w^i_{orig}, y, V_{c_y}, E_{cand})$

1: **Inputs**:
Robot i's initial information: bipartite graph $G^i_{orig} = (V, E^i_{orig})$ with bipartition (R, P), and weight function w^i_{orig}
Vertex labeling y
Minimum vertex cover $V_{c_y} = (R_{c_y}, P_{c_y})$
Bipartite graph $G = (V, E_{cand})$ containing candidate edges

2: **if** $i \in R \setminus R_{c_y}$ **then** {Robot i is uncovered, and can thus, contribute a candidate edge using its original information}

3: $\quad \delta^i = \min_{j \in P \setminus P_{c_y}} w_{orig}((i,j)) - (y(i) + y(j))$

4: $\quad E_{cand} \leftarrow E_{cand} \cup \{(i, j^\star)\}$, s.t. $w_{orig}((i,j^\star)) - (y(i) + y(j^\star)) = \delta^i, j^\star \in P \setminus P_{c_y}$

5: **end if**

6: **return** E_{cand}

Update-Label **($G_{lean}, w_{lean}, y, V_{c_y}, E_{cand}$):** Given a bipartite graph $G_{lean} = (V, E_{lean})$, where $V = (R, P)$, and $E_{lean} = (E_y, E_{cand})$, a weight function w_{lean}, a vertex labeling y, and minimum vertex cover V_{c_y}, the *Update-Label* sub-algorithm updates y as per Step 1 of the Hungarian Method (see Figures 22d and 23a for examples).

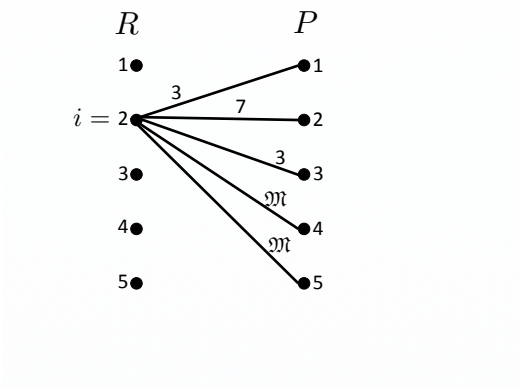
Lemma 5 *For a robot $i \in R$, the number of edges in its sparse graph $G^i_{lean} = (V, E^i_{lean})$, included in its message msg^i, is at most $|V| - 1$.*

Proof Recall that $G^i_{lean} = (V, E^i_{lean})$ is a bipartite graph, where $V = (R, P)$, and $E^i_{lean} = (E^i_y, E^i_{cand})$. Since E^i_y is constantly updated using the *Find-Matching-Cover* sub-algorithm,

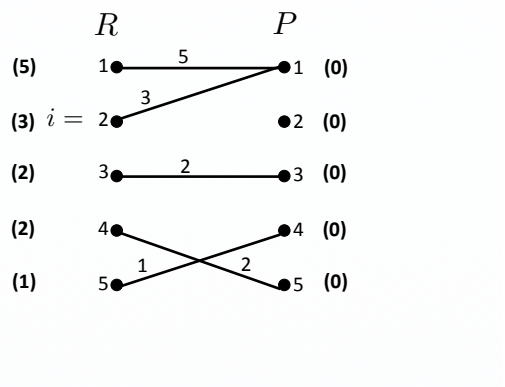
Sub-algorithm *Update-Label* $(G_{lean}, w_{lean}, y, V_{c_y}, E_{cand})$

1: **Inputs**:
Bipartite graph $G_{lean} = (V, E_{lean})$, where $V = (R, P)$, and $E_{lean} = (E_y, E_{cand})$
Weight function w_{lean}
Vertex labeling y
Minimum vertex cover $V_{c_y} = (R_{c_y}, P_{c_y})$
2: $\delta \leftarrow \min_{(i,j)\in E_{cand}} w_{lean}((i,j)) - (y(i) + y(j))$
3: Update the cover y as follows,
$y(i) \leftarrow y(i) - \delta, \forall i \in R_{c_y}$
$y(j) \leftarrow y(j) + \delta, \forall j \in \overline{P}_{c_y}$
4: $E_y \leftarrow$ equality subgraph edges, given (G_{lean}, w_{lean}) and the updated y {Using equation (68); At least one edge from E_{cand} is included in E_y}
5: **return** y, E_y

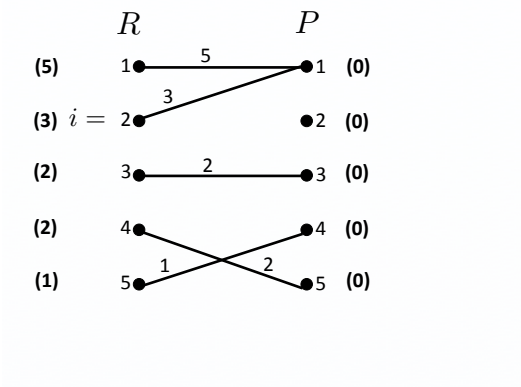
(that also generates a maximal matching M_y^i, and a corresponding minimum vertex cover $V_{c_y}^i = (R_{c_y}^i, P_{c_y}^i)$), it satisfies the following: $|E_y^i| = |R_{c_y}^i| + |V|/2$ (see point (1) in the explanation of the *Find-Matching-Cover* sub-algorithm). Moreover, from the *Get-Ecand* sub-algorithm, we know that E_{cand}^i is determined based on the same vertex cover $V_{c_y}^i$. Moreover, for each uncovered robot, E_{cand}^i contains exactly one weighted edge from that robot to some uncovered target, if such an edge exists. Thus, $|E_{cand}^i| < |R \setminus R_{c_y}^i| = |V|/2 - |R_{c_y}^i|$ (if $|E_{cand}^i| = |R \setminus R_{c_y}^i|$, then robot i has sufficient candidate edges to perform a two-step iteration of the Hungarian Method, i.e., robot i will *only* send information that it cannot update any further). Thus, $|E_{lean}^i| = |E_y^i| + |E_{cand}^i| < |R_{c_y}^i| + |V|/2 + |V|/2 - |R_{c_y}^i|$, or $|E_{lean}^i| \leq |V| - 1$. ∎



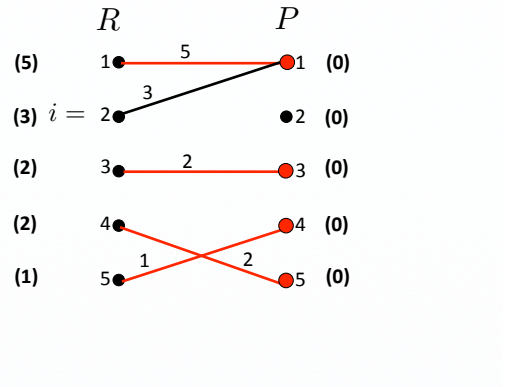
(a) Robot 2's original information in the form of the graph G^2_{orig}, and edge weight function w^2_{orig}.



(b) Robot 2's received and parsed information $G^2_{lean} = (V, E^2_{lean})$, where E^2_{lean} contains the equality sub-graph edges E^2_y (black edges) and the candidate edges E^2_{cand} (green edges, none in this case), the corresponding weight function w^2_{lean}, the vertex labeling function y^2, and the progress-counter γ^2. At this point, $\gamma^2 = 0$ and Robot 2 is ready to begin the computation phase

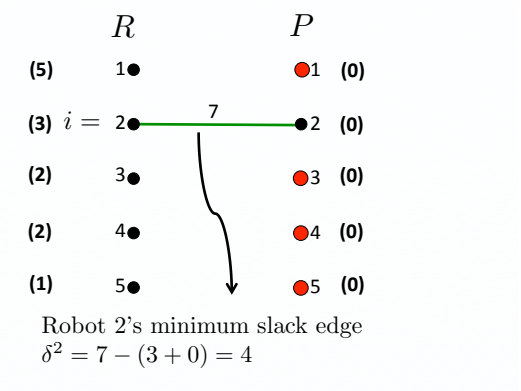


(c) The isolated set of equality sub-graph edges E^2_y from the received and parsed information.

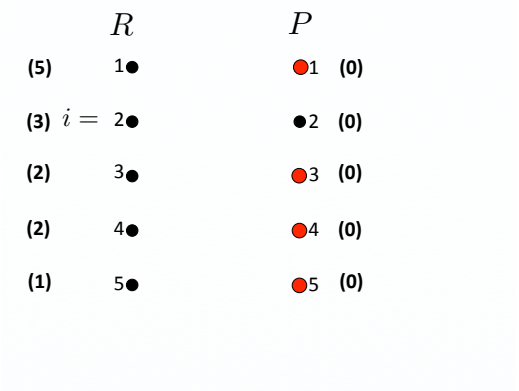


(d) For the given E^2_y, a maximum cardinality matching M^2_y (red edges), and a corresponding minimum vertex cover $V^2_{c_y} = (A^2_{c_y}, B^2_{c_y})$ (red vertices).

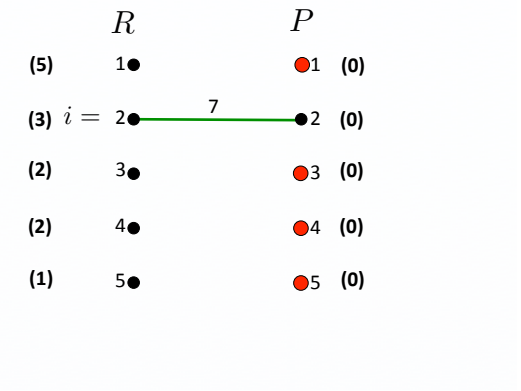
Figure 19: For the centralized example in Figure 15, an example of an instance of the *Distributed-Hungarian* algorithm, where Robot 2 performs a similar Initialization step.



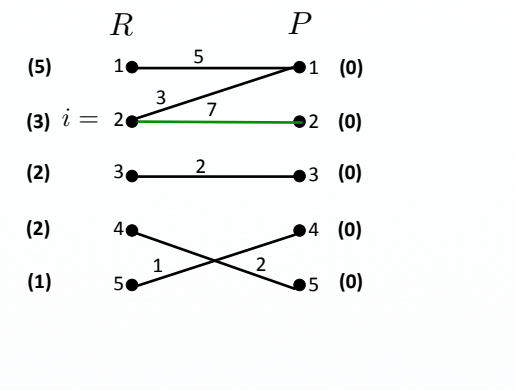
(a) For the given minimum vertex cover $V^2_{c_y}$, Robot 2's set of candidate edges (green edges, only one in this case) using its original information G^2_{orig}, and w^2_{orig} (Figure 19a). Exactly one edge (with corresponding minimum slack δ^2) is chosen for inclusion in E^2_{cand}.



(b) The isolated set of candidate edges E^2_{cand} (green edges, none in this case) from the received and parsed information (Figure 19b).

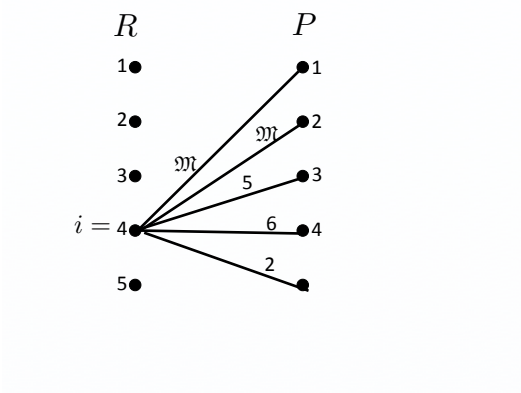


(c) The updated set of candidate edges E^2_{cand} (green edges), combining the edges from Figures 20a and 20b. As can be seen, the edges are not sufficient, i.e. an edge does not exist between *every* uncovered robot and some uncovered target.

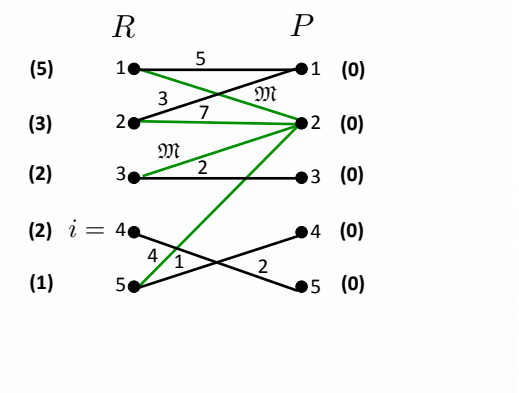


(d) Robot 2's outgoing message msg^2, comprising of the graph $G^2_{lean} = (V, E^2_{lean})$, where E^2_{lean} contains the equality sub-graph edges E^2_y (black edges) and the candidate edges E^2_{cand} (green edge), the corresponding weight function w^2_{lean}, the vertex labeling function y^2, and the progress-counter $\gamma^2 = 0$.

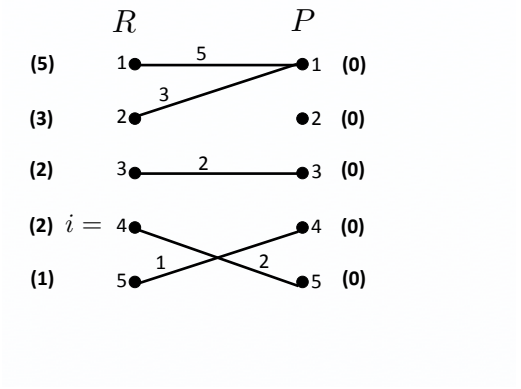
Figure 20: Continued from Figure 19, Robot 2 adds a candidate edge from its original information, to the received set of candidate edges. However, Robot 2 does not have a candidate edge from *every uncovered* robot to an uncovered target, and hence cannot perform a two-step iteration of the Hungarian Method, similar to the one depicted in the centralized example in Figure 16.



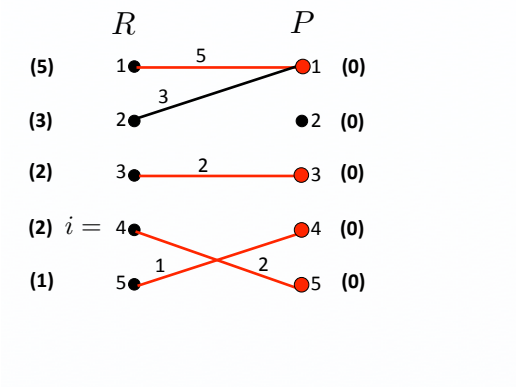
(a) Robot 4's original information in the form of the graph G^4_{orig}, and edge weight function w^4_{orig}.



(b) Robot 4's received and parsed information $G^4_{lean} = (V, E^4_{lean})$, where E^4_{lean} contains the equality sub-graph edges E^4_y (black edges) and the candidate edges E^4_{cand} (green edges), the corresponding weight function w^4_{lean}, the vertex labeling function y^4, and the progress-counter $\gamma^4 = 0$.

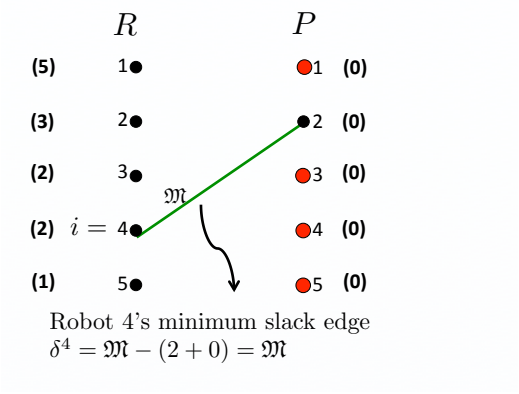


(c) The isolated set of equality sub-graph edges E^4_y from the received and parsed information.

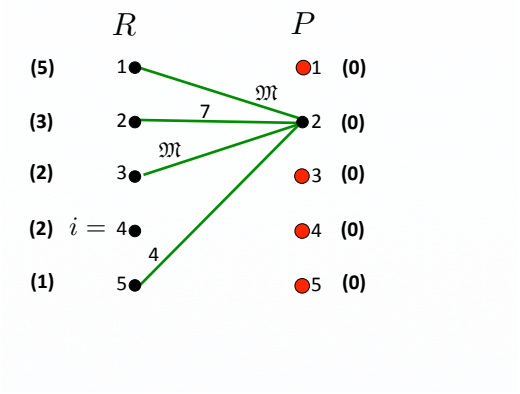


(d) For the given E^4_y, a maximum cardinality matching M^4_y (red edges), and a corresponding minimum vertex cover $V^4_{c_y} = (A^4_{c_y}, B^4_{c_y})$ (red vertices).

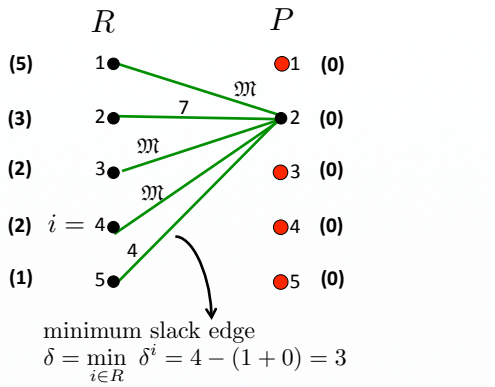
Figure 21: An example of an instance of the *Distributed-Hungarian* algorithm, where Robot 4 performs a two-step iteration of the Hungarian Method (continued through Figures 22 - 23), similar to the one depicted in the centralized example in Figure 16. Up until this point, Robot 4 performs the Initialization step.



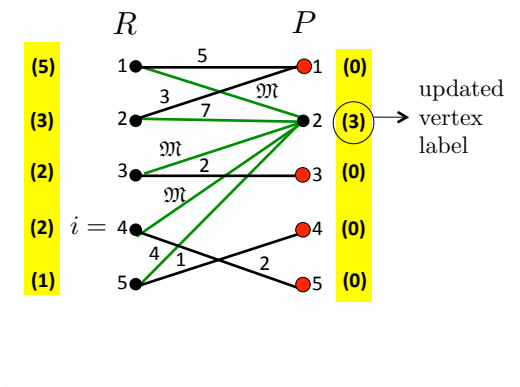
(a) For the given minimum vertex cover $V^4_{c_y}$, Robot 4's set of candidate edges (green edges) using its original information G^4_{orig}, and w^4_{orig} (Figure 21a). Exactly one edge (with corresponding minimum slack δ^4) is chosen for inclusion in E^4_{cand}.



(b) The isolated set of candidate edges E^4_{cand} (green edges) from the received and parsed information (Figure 21b).

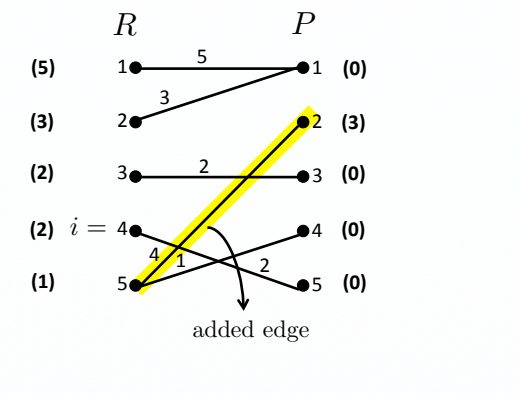


(c) Step 1: The updated set of candidate edges E^4_{cand} (green edges), combining the edges from Figures 22a and 22b. Since the edges are sufficient (there exists an edge from every uncovered robot to some uncovered target), the minimum slack δ is calculated using the edges.

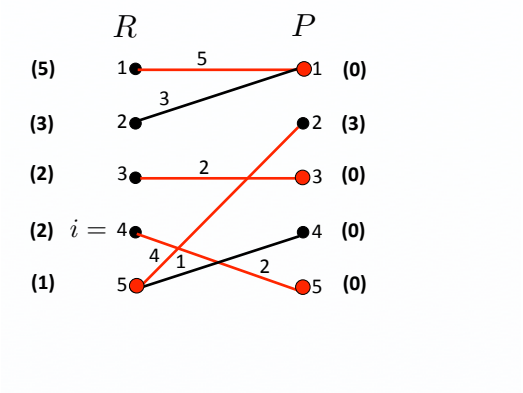


(d) Step 1: The updated vertex labeling function y^4 (highlighted in yellow), using the minimum slack δ.

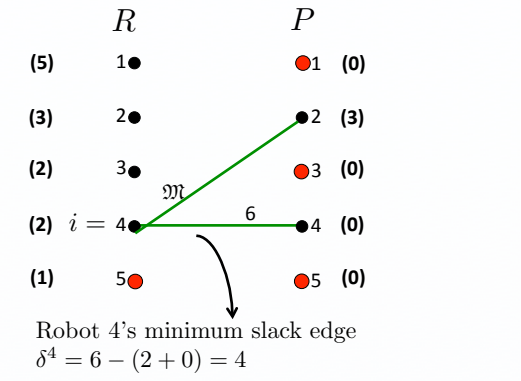
Figure 22: Continued from Figure 21, Robot 4 adds a candidate edge from its original information, to the received set of candidate edges. Since Robot 4 has sufficient candidate edges, it can perform a two-step iteration similar to the one depicted in the centralized example in Figure 16. Up until this point, Robot 4 has performed Step 1.



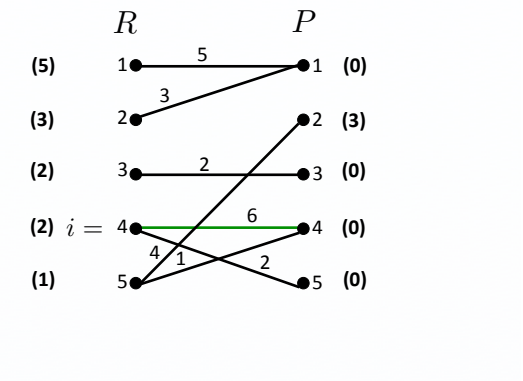
(a) Step 2: For the updated y^4, the corresponding set of equality subgraph edges E_y^4 (with the new, added edge highlighted in yellow).



(b) Step 2: For the updated E_y^4, a maximum cardinality matching M_y^4 (red edges), and a corresponding minimum vertex cover $V_{c_y}^4 = (A_{c_y}^4, B_{c_y}^4)$ (red vertices).



(c) For the updated minimum vertex cover $V_{c_y}^4$, Robot 4's set of candidate edges (green edges) using its original information G_{orig}^4, and w_{orig}^4 (Figure 21a). Exactly one edge (with corresponding minimum slack δ^4) is chosen for inclusion in E_{cand}^4



(d) Robot 4's outgoing message msg^4, comprising of the graph $G_{lean}^4 = (V, E_{lean}^4)$, where E_{lean}^4 contains the equality sub-graph edges E_y^4 (black edges) and the candidate edges E_{cand}^4 (green edge), the corresponding weight function w_{lean}^4, the vertex labeling function y^4, and the progress-counter $\gamma^4 = 1$.

Figure 23: Continued from Figure 22, Robot 4 completes the two-step iteration, and computes its outgoing message, with the updated information including a candidate edge contributed for enabling the next two-step iteration (to be performed by any robot that has sufficient information to do so).

Convergence Analysis: Recall that in the *Distributed-Hungarian* algorithm, robot i's progress-counter γ^i represents the cumulative number of *completed* stages, or two-step iterations (Steps 1 and 2) of the Hungarian Method, that results in its corresponding information given by $(G^i_{lean}, w^i_{lean}, y^i)$. In other words, we use γ^i for keeping track of the updates in the vertex labeling function y^i, and the set of corresponding equality subgraph edges E^i_y in particular, since these determine the maximum cardinality matching M^i_y, i.e. robot i's subsequent solution. Let us assume that the maximum value any robot's progress-counter can take is $\hat{k} \in \mathbb{N}_0$. Then, for $\gamma^i = k$, $k \in \{0, 1, ..., \hat{k}\}$, we denote the corresponding vertex labeling function and equality sub-graph edges by y^i_k and $E^i_{y_k}$ respectively.

We proceed to provide the main feature of the *Distributed-Hungarian* algorithm, that is key to analyzing its convergence properties:

Lemma 6 *In the Distributed-Hungarian algorithm, for every progress-counter value $k = 0, ..., \hat{k}$, there exists a unique, globally feasible*[6] *vertex labeling function, denoted by y_k, and a unique set of equality subgraph edges, denoted by E_{y_k}. In other words, during some iteration α of the algorithm, if robot i's progress-counter value equals k, then its corresponding vertex labeling function equals y_k, and the set of equality subgraph edges equals E_{y_k}, i.e.,*

$$\gamma^i = k \;\Rightarrow\; y^i_k = y_k \;\; \textit{and} \;\; E^i_{y_k} = E_{y_k}$$

Proof (by induction) By the connectivity assumption (Assumption 3), it follows that within a maximum of $|R| - 1$ iterations from the very first iteration of the *Distributed-Hungarian* algorithm, there exists at least one robot i that receives *every* robot's *initial* messages, and increments its progress-counter from -1 to 0, with an accompanying determination of a vertex labeling function y^i and corresponding set of equality subgraph edges E^i_y (Step 1: Special Case of the *Parse-Info* sub-algorithm). Let a_0 be the iteration of the algorithm, on which this occurs. Based on the *initial* messages of every robot, it is clear that y^i is globally

[6] *Feasible with respect to (G, w), i.e. the problem data for the centralized assignment problem.*

feasible.

By construction, during some iteration α, where $a_0 \leq \alpha \leq a_0 + |R| - 1$, robot j, ($j \neq i$) updates its progress-counter γ^j from -1 to 0, in one of the following ways: (i) robot j receives and operates on the exact same information used by robot i on iteration a_0, and (ii) robot j receives information from another robot with progress-counter value 0. Clearly, in both cases, $y_0^j = y_0$ (and we already showed y_0 is globally feasible), and $E_{y_0}^j = E_{y_0}$.

Without loss of generality, assume that on some iteration of the algorithm, all robots are at the above mentioned state (progress-counter values 0), and synchronously, start the computation phase of the *Distributed-Hungarian* algorithm (in reality, this need not be the case). In particular, the robots execute the Initialization step of the Hungarian Method (i.e. find identical maximal matchings, and a corresponding minimum vertex covers), and if the matchings found are not perfect, the robots begin performing the two-step iterations (Steps 1 and 2) of the Hungarian Method.

Recall that if a robot does not have sufficient candidate edges to execute Step 2, it includes (if possible) a candidate edge from its original information, to a set of candidate edges selected with respect to a minimum vertex cover associated with its equality subgraph edges, and sends such a set in its outgoing message. Clearly, there exists at least one robot p that contains a candidate edge in its original bipartite graph G_{orig}^p, that satisfies Step 1-(i) of the Hungarian Method, and can thus, be used to execute a two-step iteration, or in other words, update the progress-counter value from 0 to 1. Robot p includes such a candidate edge in its message for sending, and since all robots contribute pertinent candidate edges in context to identical minimum vertex covers, within a maximum of $(|R| - 1)$ iterations, there exists at least one robot q (may or may not be the same as p) that receives sufficient candidate edges (including the above-mentioned candidate edge that satisfies Step 1-(i)), and updates its progress-counter from 0 to 1. Let a_1 be the iteration of the algorithm, on

which the occurs. Moreover, the update satisfies Lemma 4, and hence results in a globally feasible, vertex labeling function y_1^j, and a corresponding set of equality subgraph edges E_1^j. Let $y_1 = y_1^j$, and $E_{y_1} = E_1^j$.

Following the same train of logic as before, by construction, during some iteration α, where $a_1 \leq \alpha \leq a_1 + |R| - 1$, robot s, $(s \neq q)$ updates its progress-counter γ^s from 0 to 1, in one of the following ways: (i) robot s receives and operates on the exact same information used by robot q on iteration a_1, and (ii) robot s receives information from another robot with progress-counter value 1. Clearly, in both cases, $y_1^s = y_1$ (and we already showed y_1 is globally feasible), and $E_{y_1}^s = E_{y_1}$. By assuming that for every robot $i \in R$, $\gamma^i = k$ corresponds to a unique globally feasible y_k, and a unique E_{y_k}, we can complete the proof by induction. ■

Corollary 1 *On some iteration α of the Distributed-Hungarian algorithm, if two robots have identical progress-counter values, then they have identical corresponding maximal matchings, i.e.,*

$$\gamma^i = \gamma^j \Rightarrow M^i = M^j \ \forall i, j \in R, i \neq j$$

Proof Let $\gamma^i = \gamma^j = k$, for some $k \in \{0, 1, ..., \hat{k}\}$. From Lemma 6, we know that the progress-counter value k corresponds to a unique set of equality subgraph edges E_{y_k}, and consequently, a unique maximal matching. Let such a matching be denoted by M_{y_k}, where $M_{y_k} \subseteq E_{y_k}$). Thus, robots i and j have identical maximal matchings, equal to M_{y_k} corresponding to the progress-counter value k. ■

Corollary 2 *On some iteration α of the Distributed-Hungarian algorithm, if robot i's progress-counter value is higher than robot j's progress-counter value, i.e. $\gamma^i > \gamma^j$, then one of the following is true,*

(i) Robot i's maximal matching is greater in size than robot j's maximal matching, i.e. $|M^i| > |M^j|$, or

(ii) Robots i and j have the same maximal matching, i.e. $M^i = M^j$, but $|R^i_{c_y}| > |R^j_{c_y}|$, i.e. robot i has more covered vertices in R than robot j.

Proof Let $\gamma^i = p$, and $\gamma^j = q$, where $p, q \in \{0, 1, ..., \hat{k}\}$, $p > q$. From Lemma 6, it is clear that for every $k \in \{q+1, ..., p-1\}$, there *existed* a robot with progress-counter value k, and corresponding globally feasible vertex labeling function y_k, equality subgraph edges E_{y_k}, and maximal matching M_{y_k}, during some previous iteration of the algorithm. Thus, we can use Lemma 4 iteratively, for every progress-counter update that occurred from k to $k+1$, $k \in \{q, q+1, ..., p-1\}$, to show that either statement (i) or (ii) from Lemma 4 is true (since each progress-counter update corresponds to a two-step iteration of the Hungarian Method). ■

Theorem 6 *Given a set of robots R, a set of targets P, and a time-varying communication graph $\mathcal{G}_c(t)$ as per Assumption 3. Assuming every robot $i \in R$ knows (R, P, c^i), if the robots execute the Distributed-Assignment algorithm, there exists a finite number of iterations such that all robots converge to identical optimal solutions $\hat{M} \in \mathcal{M}$, that are also optimal to the corresponding centralized assignment problem.*

Proof We consider the following two cases: On some iteration α of the *Distributed-Hungarian* algorithm, either (i) there exists at least one robot i such that $\gamma^i \geq \gamma^j$, $\forall j \in R$, $j \neq i$, or (ii) all robots have the same progress-counter values, i.e. $\gamma^i = \gamma^j$, $\forall i, j \in R$.

In the first case, using the connectivity assumption (Assumption 3), within a maximum of $(|R| - 1)$ iterations, all robots receive information from robot i, and by construction, update their information to robot i's information, bringing us to the second case: Suppose $\gamma^i = \gamma^j = k$, $\forall i, j \in R$, where $k \in \{0, 1, ..., \hat{k}\}$. Using Corollary 1, we know that all robots have identical maximal matchings, equal to M_{y_k}. Clearly, if M_{y_k} is perfect, then $k = \hat{k}$ (since the progress-counter ceases to update, once a perfect matching is found). Moreover, $M_{y_{\hat{k}}}$ is optimal. This is true because we know from Lemma 6, that $y^i_{\hat{k}}$ is globally

feasible, and since the maximal matching is contained within the equality subgraph edges, i.e. $M_{y_{\hat{k}}} \subseteq E_{y_{\hat{k}}}$, from Theorem 5, we can conclude that $M_{y_{\hat{k}}}$ is optimal.

Now, suppose in the second case, M_{y_k} is not perfect (and hence, not optimal). Then, from the proof of Lemma 6, we know that within a maximum of $(|R|-1)$ iterations, there exists at least one robot that executes an update, i.e. a two-step iteration of the Hungarian Method with a consequent increment in its progress-counter. Moreover, from Lemma 4, we know that there is a finite number of such updates that can occur (worst case $|V|^2$), before a robot finds a perfect matching. ∎

Detecting Infeasibility: If the centralized assignment problem (G, w) is infeasible, the *Distributed-Hungarian* algorithm converges to a matching $\hat{M}$ that contains edges with infeasible weights (i.e. denoted by $\mathfrak{M}$ as per the big-M method).

Corollary 3 (Stopping Criterion) *Suppose robot i finds a perfect matching M_y^i during some iteration α of the Distributed-Hungarian algorithm. Then robot i can stop sending its corresponding message $msg^i = (G_{lean}^i, w_{lean}^i, y^i)$ (where $E_{lean}^i = (\hat{M}, \emptyset)$) after $(|R|-1)$ iterations.*

Proof The above result is derived from the following fact: if the matching M_y^i, found by robot i is perfect, then it is also optimal (as discussed in the proof of Theorem 6). Moreover, no other robot can find a perfect (and hence optimal) matching, different from M_y^i, i.e. $M_y^i = \hat{M}$ (using Lemma 6). Thus, within a maximum of $(|R|-1)$ iterations, every robot in the network will receive robot i's message, and update its own information to robot i's solution, at which point, robot i need not send its message anymore. ∎

Corollary 4 *All robots converge to identical optimal solutions $\hat{M} \in \mathcal{M}$ in $\mathcal{O}(|V|^3)$ iterations.*

Proof As discussed previously, the Hungarian Method converges to an optimal solution in $\mathcal{O}(|V|^2)$ two-step iterations. In context to the *Distributed-Hungarian* algorithm, each

two-step iteration of the Hungarian Method corresponds to a progress-counter update that occurs within $\mathcal{O}(|V|)$ iterations of the *Distributed-Hungarian* algorithm, and thus, every robot converges to $\hat{M}$ in $\mathcal{O}(|V|^3)$ iterations. ∎

Lemma 7 *On every iteration of the Distributed-Hungarian algorithm, a robot transmits* $\mathcal{O}(|V|log|V|)$ *bytes.*

Proof Recall that a robot i's message, msg^i, comprises of a sparse graph $G^i_{lean} = (V, E^i_{lean})$ with at most $|V| - 1$ edges (see Lemma 5), a weight function $w^i : E^i_{lean} \rightarrow \mathbb{R}$, a vertex labeling function $y^i : V \rightarrow \mathbb{R}$, and a progress-counter value, $\gamma^i \in \{0, 1, ..., \hat{k}\}$, where $\hat{k}$ denotes the maximum number of two-step iterations of the Hungarian Method, and is of the order of $\mathcal{O}(|V|^2)$ (using Lemma 4). Note that we can encode each edge in E_{lean}, in $1/4 \log_2(|V|/2)$ bytes. Moreover, by approximating a real number as a 16-bit floating point value, we can encode the corresponding edge weights in w^i_{lean}, and vertex labels in y^i, in $2(|V| - 1)$ and $2|V|$ number of bytes respectively. Additionally, we can encode the progress-counter value in the order of $\mathcal{O}(2 \log_2(|V|))$. Thus, robot i's message, msg^i, comprises of $\mathcal{O}(|V|log|V|)$ bytes. ∎

Complexity of the Distributed-Hungarian algorithm: The run time of a single iteration of the *Distributed-Hungarian* algorithm is $\mathcal{O}(r^2)$. In contrast to the centralized Hungarian Method, where a solution is reached in $\mathcal{O}(r^3)$, here every robot executes sub-steps of the Hungarian Method, and hence computes *less* per iteration of the distributed algorithm, i.e. every robot takes $\mathcal{O}(r^2)$ time per iteration of the algorithm. Since the robots compute identical optimal solutions in $\mathcal{O}(r^3)$ iterations, the overall runtime of the *Distributed-Hungarian* algorithm is $\mathcal{O}(r^5)$.

5.2 Distributed Dynamic Spatio-Temporal Routing

In this section, we discuss high level concepts that extend our work on distributed assignments, to spatio-temporal routing. So far, in previous chapters, we dealt with finding routes for robots given a static (fixed) Score, where the routes were obtained in a centralized manner *beforehand* (i.e. the robots executed pre-determined routes). Now we consider spatio-temporal routing under a *distributed* framework, where the robots determine routes *online*, given a *dynamic* Score. By dynamic, we mean that the Score (comprising of timed positions) can change in real-time, e.g. through user involvement, and the robots are required to adapt to the changes. We incorporate such a dynamic aspect by broadcasting the changes made to the Score, that the robots then assimilate while determining their routes.

In the following paragraph, we put forth the two key ideas, central to the execution of the above mentioned distributed dynamic spatio-temporal routing problem. For convenience, we assume the following:

Assumption 4 *Let t_{pos} be the minimum time between consecutive timed positions in the Score. Additionally, let t_{asgn} be the worst-case time required to solve the Distributed ("one-step") Assignment Problem from Section 5.1.2, using the proposed Distributed-Hungarian algorithm. Then, (i) $t_{pos} \geq t_{asgn}$, and (ii) choose the initial time instant t_0 such that $(t_1 - t_0) \geq t_{asgn} + t_{pos}$.*

The above assumption ensures that every timed position in the Score is assigned to a robot, at a time instant *before* it needs to be reached by that robot.

- *Distributed Aspect:* Given a Score, the robots determine routes by iteratively solving assignments between successive time instants, employing the *Distributed-Hungarian* algorithm for each assignment. Moreover, the robots solve assignments between *future* consecutive time instants, while *simultaneously* executing routes that they have already determined (see Figure 24) (We assume that the underlying time-varying communication

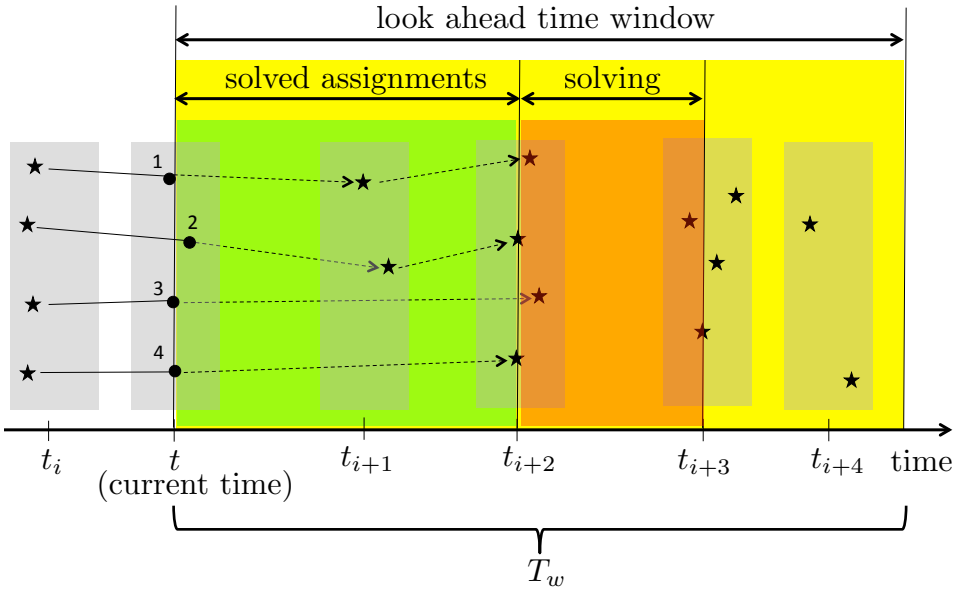
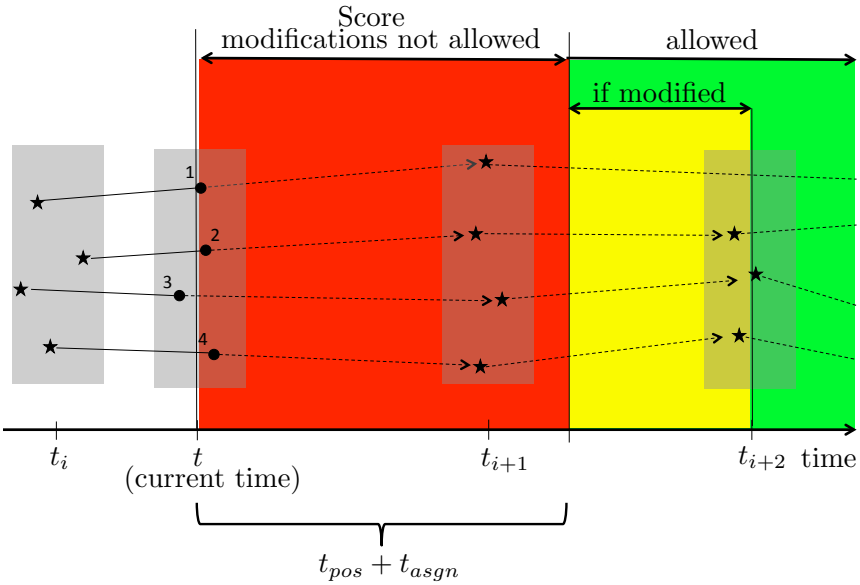


Figure 24: Robots (circles) at time t with a look-ahead window T_w, and timed positions (stars) grouped in grey boxes representing specific time instant, where the robots are currently solving an assignment between *future* consecutive time instants (t_{i+2} - t_{i+3}), while executing routes they have already determined (t_i - t_{i+1}).

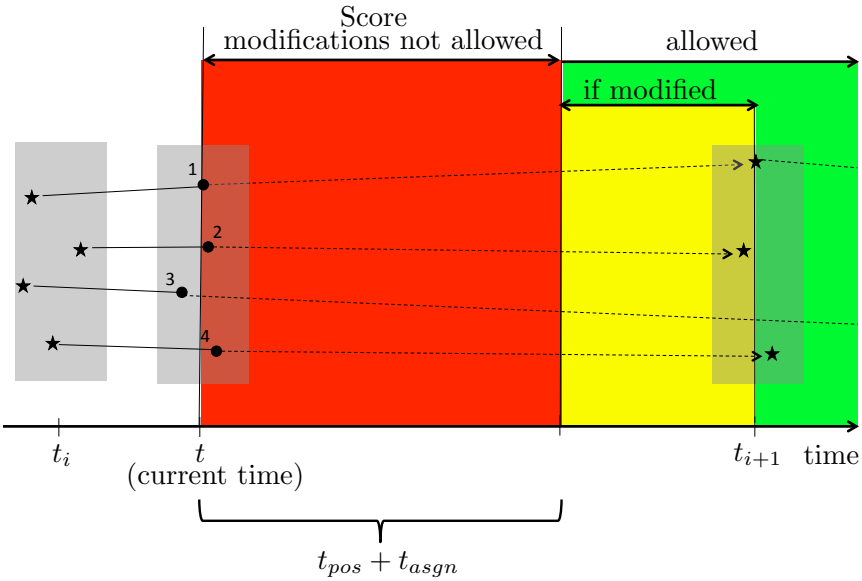
graph, induced as the robots execute their paths, satisfies Assumption 3). Note that the robots may have a moving look-ahead time window $T_w \in \mathbb{R}_{\geq 0}$, constraining the number of future timed positions that they have knowledge of, thus constraining how far-ahead in time, they can plan their routes.

- *Dynamic Aspect:* Modifications made to the Score must satisfy three criteria before they are broadcasted to the robots: first, the changes should not violate feasibility (in that, the available number of robots should not fall short)[7], second, all changes should be specified at time instants later than $t + (t_{pos} + t_{asgn})$, where t denotes the current time, to allow the robots a certain minimum time interval to respond to the changes (see Figure 25), and third, all addition-specific changes must respect the minimum time constraint (δ_{min}) between consecutive timed positions.

[7]Velocity constraints are not considered here, since the user has no knowledge of the positions of the robots, and hence, cannot ascertain feasibility in that respect.



(a) Case 1: If at the current time t, a modification is received, specified at a time instant in the depicted time interval (yellow), *all* robots retain their routes up until time instant t_{i+1} and begin recalculating their routes, from their positions at t_{i+1} onwards.



(b) Case 2: If at the current time t, a modification is received, specified at a time instant in the depicted time interval (yellow), *all* robots begin recalculating their routes, from their current positions at time t onwards.

Figure 25: Robots (circles) at time t, and timed positions (stars) grouped in grey boxes representing specific time instant, with depictions of the time periods in which a user cannot (red) and can (green) modify the Score.

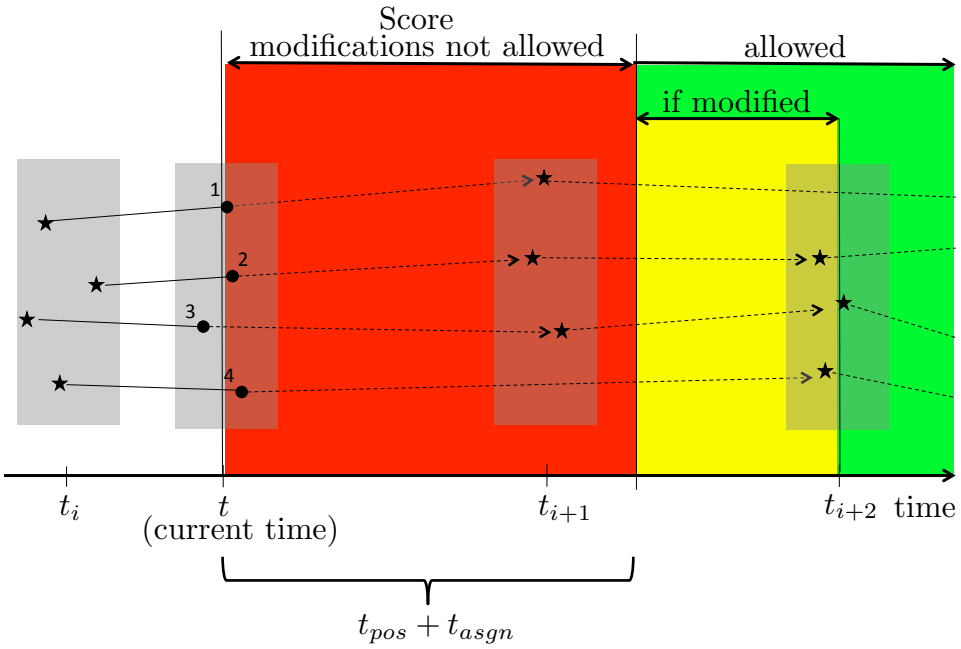


Figure 26: Continued from Figure 25, Case 3: If at the current time t, a modification is received, specified at a time instant in the depicted time interval (yellow), *only* robots $1, 2$ and 3 retain their routes up until time instant t_{i+1}. The robots begins recalculating their routes from their positions at t_{i+1} onwards, however robot 4 begins recalculating its route from its current position at time t onwards.

We provide a brief description of the process of incorporating dynamic modifications, in the determination of routes. Since the routes of the robots are determined through piece-wise assignments between robot positions and timed positions at successive time instants in the Score, the instant a dynamic modification is received, each robot chooses the time instant in the Score up until which, its previously-determined routes need not be modified, and begins recalculating its route *from such a time instant onwards* (while executing its trajectory on the previously determined route). In Figures 25a, 25b and 26, we provide examples of three different cases that can occur, when a particular dynamic modification is received.

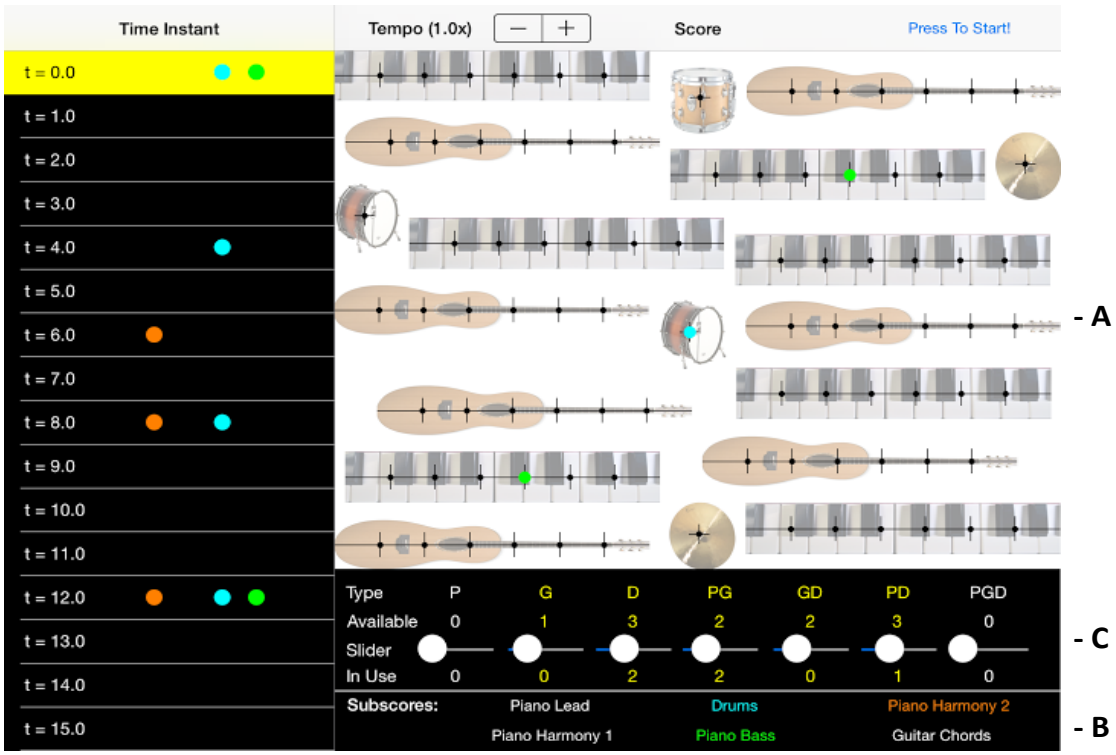


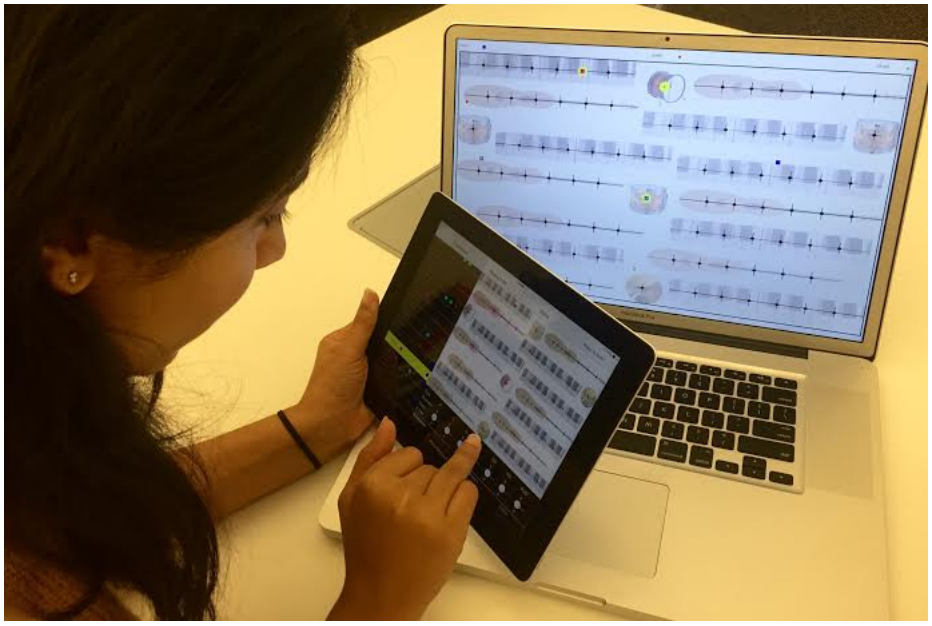
Figure 27: An iPad interface for modifying and broadcasting changes to the Score. **A**: The Robot Music Wall 3.0 comprising of unique positions corresponding to piano and guitar notes, as well as drum beats. **B**: Pre-defined sub-scores for an instrumental version of "The Final Countdown". **C**: A feature for either selecting, or setting the available number of robots per skill set.

5.2.1 Simulation and Hardware Results (Robot Music Wall 3.0)

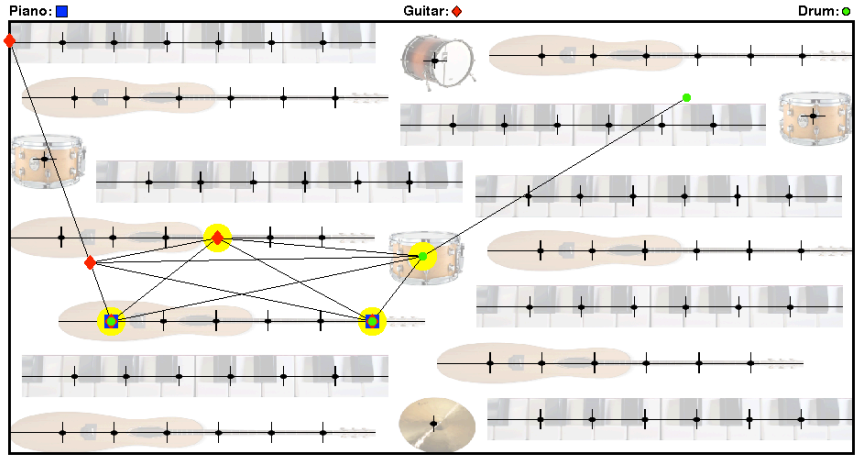
In this section, we introduce the third and final version of the Robot Music Wall, developed in MATLAB and instrumented to include a larger array of piano, guitar and drum sounds (see A in Figure 27). In addition to the Wall, we developed a graphical user-interface (GUI) (Figure 27) that allows a user to create, and administer changes to a musical Score on the Wall. The user-interface is developed on an iPad that broadcasts the changes issued by a user, to the robots executing the Score. For convenience, we created beforehand, a heterogeneous Score comprising of piano and guitar notes, and drum beats associated with the popular song "The Final Countdown" by the Swedish band "Europe".

We divided the Score into multiple single-instrument sub-scores. For instance, we separated the piano notes into individual sub-scores corresponding to the piano lead, piano bass, and second and third harmonies (see B in Figure 27). The motivation behind the creation of such sub-scores was to enable a user to "add, delete or modify" the Score through these structures, in an intuitive and immediately recognizable manner. In addition to the sub-scores, we included the option of adding and removing individual timed positions, and switching instruments (pianos to guitars and vice versa, drums from one type to another).

To execute an example of dynamic distributed spatio-temporal routing, the user selects the number of robots available for use, and creates a Score using the iPad interface. Once the user hits the start button, the iPad broadcasts this Score to the robots (simulated on a separate computer, see Figure 28). From this point onwards, the robots determine and execute routes in real-time, while the iPad broadcasts changes to the Score, as and when a user decides to modify it. We implemented a real-time rendition of "The Final Countdown" on the basis of the above mentioned method, in both simulation and hardware environments (Figures 28 and 29 respectively).

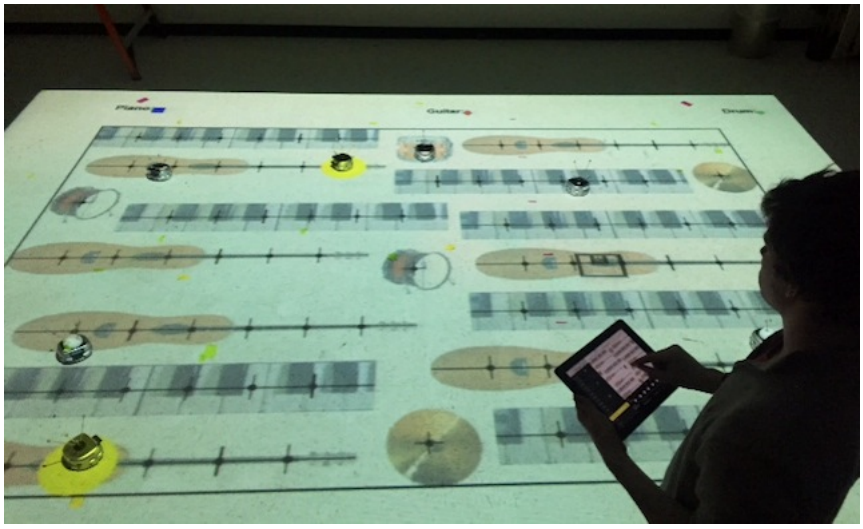


(a) A user interacting with simulated robots through the music wall iPad interface.

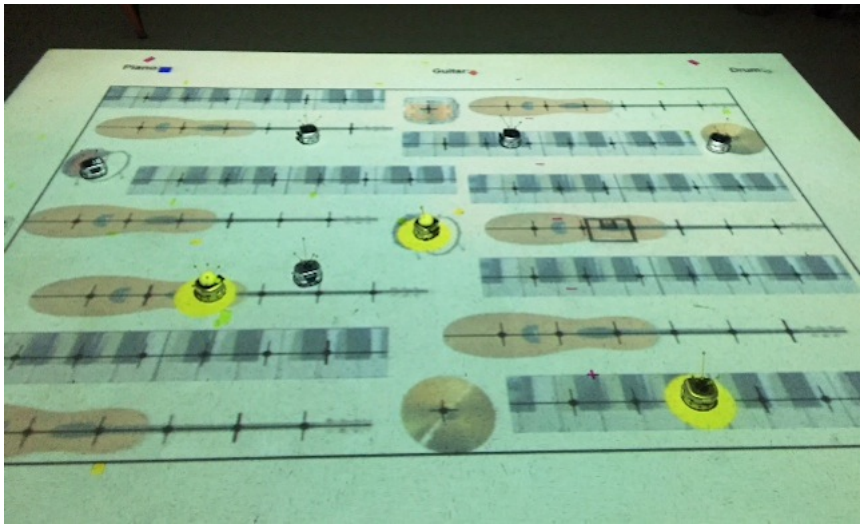


(b) An instance of simulated robots (geometric shapes - diamonds, circles and squares), performing the Score by executing dynamic spatio-temporal routing (the black lines denote the underlying dynamic communication network, required for a distributed implementation).

Figure 28: Simulation of multiple robots performing “The Final Countdown” on the Wall, with a user “conducting” (modifying) the Score through the iPad interface.



(a) A user interacting with actual robots through the music wall iPad interface.



(b) An instance of actual robots, performing the Score by executing dynamic spatio-temporal routing.

Figure 29: Hardware implementation of multiple robots performing "The Final Countdown" on the Wall, with a user "conducting" (modifying) the Score through the iPad interface.

CHAPTER VI

CONCLUDING REMARKS

The research presented in this dissertation can be used towards customizing and comparing various instances of spatio-temporal routing, for example centralized versus distributed versions, static versus dynamic Scores, or homogeneous versus heterogeneous capabilities. In subsequent paragraphs, we highlight three aspects of our work that are not explicitly addressed in this document, but are pertinent to the implementation of the concepts provided, onto a practical hardware platform.

1. Collision Avoidance: In the homogeneous case (where robots and timed positions in the Score are *not* associated with skill sets), minimum distance paths in $\mathbb{R}^2$ *do not intersect*, and as a result, we are guaranteed routes that do not cross one another. However, the same does not apply to the heterogeneous case. Moreover, since the robots are not point particles in the real world, we should incorporate collision avoidance in the lower level controller that the robots use to execute their routes, for both homogeneous as well as heterogeneous cases.

2. Velocity Constraints: Though we explicitly account for a maximum velocity that the robots cannot exceed, in the analysis of spatio-temporal routing in Section 3.1, we do not incorporate this constraint *in conjunction with* connectivity maintenance in Section 3.2 or heterogeneity in Chapter 4 (due to the prohibitive complexity of the ensuing optimization problems). However, since we provide heuristic algorithms for generating the routes of the robots in both cases, we can extend our work using techniques similar to the ones proposed in Section 3.1, to incorporate a maximum velocity, thereby yielding a lower bound on the minimum number of robots required,

and consequent routes that do not violate the velocity constraint.

3. Robot Failure: Though we do not explicitly implement robustness to robot failure, it is conceivable that if a robot fails during the distributed execution of spatio-temporal routing, and the failure is detected by another (neighboring) robot, the information about such a failure can be eventually communicated to all robots, and they can exclude the robot in question from future computations.

In conclusion, through this dissertation, we presented an analysis of spatio-temporal routing under various constraints specific to multi-robot applications. We began with the basic version of spatio-temporal routing, called the Unconstrained Routing Problem (URP), and provided insight into how the framework of assignment problems could be applied towards finding solutions. Next, we analyzed the URP under interesting practical constraints like maximum robot velocities and network connectivity, followed by a further generalization involving robots with heterogeneous capabilities. For each of these cases, we provided results on feasibility and route generation, using techniques drawn from the theory of assignment problems. To analyze our work in context to a distributed framework, we developed a novel, distributed version of the Hungarian Method used for solving assignment problems. With this algorithm in hand, we extended spatio-temporal routing to its distributed dynamic counterpart. Using the medium of music, we demonstrated the entirety of our research through simulations, hardware implementations and graphical user-interfaces, centered around the concept of the Robot Music Wall.

REFERENCES

[1] ALIGHANBARI, M. and HOW, J., "Decentralized task assignment for unmanned aerial vehicles," in European Control Conference, pp. 5668–5673, 2005.

[2] ARORA, S. and PURI, M., "A variant of time minimizing assignment problem," European Journal of Operational Research, vol. 110, no. 2, pp. 314 – 325, 1998.

[3] BALINSKI, M., "Signature methods for the assignment problem," Operations research, vol. 33, no. 3, pp. 527–536, 1985.

[4] BEKTAS, T., "The multiple traveling salesman problem: an overview of formulations and solution procedures," Omega, vol. 34, no. 3, pp. 209–219, 2006.

[5] BERTSEKAS, D., "A new algorithm for the assignment problem," Mathematical Programming, vol. 21, no. 1, pp. 152–171, 1981.

[6] BERTSEKAS, D., "The auction algorithm for assignment and other network flow problems: A tutorial," Interfaces, vol. 20, no. 4, pp. 133–149, 1990.

[7] BODIN, L., GOLDEN, B., ASSAD, A., and BALL, M., "Routing and scheduling of vehicles and crews: The state of the art.," Computers and Operations Research, vol. 10, no. 2, pp. 63–211, 1983.

[8] BRUMITT, B. and STENTZ, A., "Grammps: A generalized mission planner for multiple mobile robots in unstructured environments," in IEEE International Conference on Robotics and Automation, vol. 2, pp. 1564–1571, 1998.

[9] BULLO, F., FRAZZOLI, E., PAVONE, M., SAVLA, K., and SMITH, S., "Dynamic vehicle routing for robotic systems," Proceedings of the IEEE, vol. 99, no. 9, pp. 1482 –1504, 2011.

[10] BURGARD, W., MOORS, M., STACHNISS, C., and SCHNEIDER, F., "Coordinated multi-robot exploration," IEEE Transactions on Robotics, vol. 21, no. 3, pp. 376 – 378, 2005.

[11] BÜRGER, M., NOTARSTEFANO, G., BULLO, F., and ALLGÖWER, F., "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," Automatica, vol. 48, no. 9, pp. 2298–2304, 2012.

[12] BURKARD, R., DELL'AMICO, M., and MARTELLO, S., Assignment problems. Society for Industrial and Applied Mathematics, 2009.

[13] CHANG, G. and HO, P.-H., "The beta-assignment problems," European Journal of Operational Research, vol. 104, no. 3, pp. 593 – 600, 1998.

[14] CHEN, D., DU, D., HU, X., LIN, G., WANG, L., and XUE, G., "Approximations for steiner trees with minimum number of steiner points," Journal of Global Optimization, vol. 18, pp. 17–33, 2000.

[15] CHENG, X., DU, D., WANG, L., and XU, B., "Relay sensor placement in wireless sensor networks," Wireless Networks, vol. 14, pp. 347–355, 2008.

[16] CHOPRA, S. and EGERSTEDT, M., "Multi-robot routing for servicing spatio-temporal requests: A musically inspired problem," Proceedings of the IFAC Conference on Analysis and Design of Hybrid Systems, 2012.

[17] CHOPRA, S. and EGERSTEDT, M., "Multi-robot routing under connectivity constraints," 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems, 2012.

[18] CHOPRA, S. and EGERSTEDT, M., "Heterogeneous multi-robot routing," Proceedings of the American Control Conference, 2014.

[19] CHOPRA, S. and EGERSTEDT, M., "Spatio-temporal multi-robot routing," Automatica, 2014. Accepted for Publication. To appear 2015.

[20] CHOPRA, S., RICE, M., and EGERSTEDT, M., "A multi-robot orchestra: dynamic spatio-temporal routing," in IEEE Conference on Inteligent Robots and Systems, 2015. Submitted.

[21] DERIGS, U., "The shortest augmenting path method for solving assignment problems - motivation and computational experience," Annals of Operations Research, vol. 4,

pp. 57–102, 1985.

[22] DIAS, M., ZLOT, R., KALRA, N., and STENTZ, A., "Market-based multirobot coordination: A survey and analysis," Proceedings of the IEEE, vol. 94, no. 7, pp. 1257–1270, 2006.

[23] DIONNE, D. and RABBATH, C., "Multi-uav decentralized task allocation with intermittent communications: The dtc algorithm," in Proceedings of the American Control Conference, pp. 5406–5411, 2007.

[24] DIXON, C. and FREW, E., "Maintaining optimal communication chains in robotic sensor networks using mobility control," Mobile Networks and Applications, vol. 14, pp. 281–291, 2009.

[25] DUTTA, H. and KARGUPTA, H., "Distributed linear programming and resource management for data mining in distributed environments," in IEEE Data Mining Workshops, pp. 543–552, 2008.

[26] FRANCESCHELLI, M., ROSA, D., SEATZU, C., and BULLO, F., "Gossip algorithms for heterogeneous multi-vehicle routing problems," Nonlinear Analysis: Hybrid Systems, vol. 10, pp. 156 – 174, 2013.

[27] GARFINKEL, R., "An improved algorithm for the bottleneck assignment problem," Operations Research, pp. 1747–1751, 1971.

[28] GERKEY, B. and MATARIC, M., "Sold!: Auction methods for multirobot coordination," Robotics and Automation, IEEE Transactions on, vol. 18, no. 5, pp. 758–768, 2002.

[29] GERKEY, B. and MATARIĆ, M., "A formal analysis and taxonomy of task allocation in multi-robot systems," The International Journal of Robotics Research, vol. 23, no. 9, pp. 939–954, 2004.

[30] GIORDANI, S., LUJAK, M., and MARTINELLI, F., "A distributed algorithm for the multi-robot task allocation problem," in Trends in Applied Intelligent Systems, pp. 721–730, Springer, 2010.

[31] GOLDENBERG, D. K., LIN, J., MORSE, A. S., ROSEN, B. E., and YANG, Y. R., "Towards mobility as a network control primitive," in Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, pp. 163–174, 2004.

[32] GRAHAM, R. and HELL, P., "On the history of the minimum spanning tree problem," Annals of the History of Computing, vol. 7, no. 1, pp. 43–57, 1985.

[33] GUPTA, S. and PUNNEN, A., "Minimum deviation problems," Operations research letters, vol. 7, no. 4, pp. 201–204, 1988.

[34] GUTIN, G. and PUNNEN, A., The traveling salesman problem and its variations, vol. 12. Springer, 2002.

[35] HOPCROFT, J. and KARP, R., "An nˆ5/2 algorithm for maximum matchings in bipartite graphs," SIAM Journal on computing, vol. 2, no. 4, pp. 225–231, 1973.

[36] HOPCROFT, J., SCHWARTZ, J., and SHARIR, M., "On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem"," The International Journal of Robotics Research, vol. 3, no. 4, pp. 76–88, 1984.

[37] HOWARD, A., PARKER, L., and SUKHATME, G., "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection," The International Journal of Robotics Research, vol. 25, no. 5-6, pp. 431–447, 2006.

[38] JI, M., AZUMA, S.-I., and EGERSTEDT, M., "Role-assignment in multi-agent coordination," 2006.

[39] KAN, Z., DANI, A., SHEA, J., and DIXON, W., "Information flow based connectivity maintenance of a multi-agent system during formation control," Proceedings of the IEEE Conference on Decision and Control, pp. 2375 –2380, 2011.

[40] KARP, R., "Reducibility among combinatorial problems," Complexity of Computer Computations, pp. 85–103, 1972.

[41] KOES, M., NOURBAKHSH, I., and SYCARA, K., "Heterogeneous multirobot coordination with spatial and temporal constraints," Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI), pp. 1292–1297, 2005.

[42] KONIG, D., "Gráfok és mátrixok. matematikai és fizikai lapok, 38: 116–119, 1931."

[43] KOOPMANS, T. C. and BECKMANN, M., "Assignment problems and the location of economic activities," Econometrica: journal of the Econometric Society, pp. 53–76, 1957.

[44] KUHN, H., "The hungarian method for the assignment problem," Naval Research Logistics, vol. 2, no. 1-2, pp. 83–97, 1955.

[45] LAGOUDAKIS, M., MARKAKIS, E., KEMPE, D., KESKINOCAK, P., KLEYWEGT, A., KOENIG, S., TOVEY, C., MEYERSON, A., and JAIN, S., "Auction-based multi-robot routing," in Robotics: Science and Systems, pp. 343–350, 2005.

[46] LAWLER, E., Combinatorial optimization: Networks and matroids. Holt, Rinehart and Winston, New York, 1976.

[47] LERMAN, K., JONES, C., GALSTYAN, A., and MATARIĆ, M. J., "Analysis of dynamic task allocation in multi-robot systems," The International Journal of Robotics Research, vol. 25, no. 3, pp. 225–241, 2006.

[48] LIN, G. and XUE, G., "Steiner tree problem with minimum number of steiner points and bounded edge-length," Information Processing Letters, vol. 69, no. 2, pp. 53 – 57, 1999.

[49] MARTELLO, S. and TOTH, P., "Linear assignment problems," Annals of Discrete Mathematics, vol. 31, pp. 259–282, 1987.

[50] MARTELLO, S. and TOTH, P., "The bottleneck generalized assignment problem," European Journal of Operational Research, vol. 83, no. 3, pp. 621 – 638, 1995.

[51] NGUYEN, H., PEZESHKIAN, N., RAYMOND, M., GUPTA, A., and SPECTOR, J., "Autonomous communication relays for tactical robots," Proceedings of the 11th International Conference on Advanced Robotics, 2003.

[52] OLFATI-SABER, R., "Near-identity diffeomorphisms and exponential epsilon-tracking and epsilon-stabilization of first-order nonholonomic se(2) vehicles," Proceedings of the American Control Conference, vol. 6, pp. 4690 – 4695, 2002.

[53] PAPADIMITRIOU, C. and STEIGLITZ, K., Combinatorial optimization: algorithms and complexity. Courier Dover Publications, 1998.

[54] PAVONE, M. and FRAZZOLI, E., "Dynamic vehicle routing with stochastic time constraints," IEEE International Conference on Robotics and Automation, pp. 1460 – 1467, 2010.

[55] PENTICO, D., "Assignment problems: A golden anniversary survey," European Journal of Operational Research, vol. 176, no. 2, pp. 774 – 793, 2007.

[56] PEREIRA, G., DAS, A., KUMAR, R., and CAMPOS, M., "Decentralized motion planning for multiple robots subject to sensing and communication constraints," Proceedings of the Second Multi-Robot Systems Workshop, pp. 267–278, 2003.

[57] PIMENTA, L., KUMAR, V., MESQUITA, R., and PEREIRA, G., "Sensing and coverage for a network of heterogeneous robots," in IEEE Conference on Decision and Control, pp. 3947–3952, 2008.

[58] PONDA, S., JOHNSON, L., CHOI, H., and HOW, J., "Ensuring network connectivity for decentralized planning in dynamic environments," AIAA Aerospace Information Technologies Conference, 2011.

[59] PONDA, S., REDDING, J., CHOI, H.-L., HOW, J. P., VAVRINA, M., and VIAN, J., "Decentralized planning for complex missions with dynamic communication constraints," in Proceedings of the American Control Conference, pp. 3998–4003, 2010.

[60] RALPHS, T., "Parallel branch and cut for capacitated vehicle routing," Parallel Computing, vol. 29, no. 5, pp. 607 – 629, 2003.

[61] RAMCHURN, S., POLUKAROV, M., FARINELLI, A., TRUONG, C., and JENNINGS, N., "Coalition formation with spatial and temporal constraints," in Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, vol. 3,

pp. 1181–1188, 2010.

[62] RYAN, J., BAILEY, T., MOORE, J., and CARLTON, W., "Reactive tabu search in unmanned aerial reconnaissance simulations," in Proceedings of the 30th conference on Winter simulation, pp. 873–880, 1998.

[63] SARIEL-TALAY, S., BALCH, T. R., and ERDOGAN, N., "Multiple traveling robot problem: A solution based on dynamic task selection and robust execution," Mechatronics, IEEE/ASME Transactions on, vol. 14, no. 2, pp. 198–206, 2009.

[64] SARIEL-TALAY, S., BALCH, T. R., and ERDOGAN, N., "A generic framework for distributed multirobot cooperation," Journal of Intelligent & Robotic Systems, vol. 63, no. 2, pp. 323–358, 2011.

[65] SCERRI, P., FARINELLI, A., OKAMOTO, S., and TAMBE, M., "Allocating tasks in extreme teams," in Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems, pp. 727–734, 2005.

[66] SCHNEIDER, J., APFELBAUM, D., BAGNELL, D., and SIMMONS, R., "Learning opportunity costs in multi-robot market based planners," in IEEE International Conference on Robotics and Automation, pp. 1151–1156, 2005.

[67] SCHRIJVER, A., Theory of linear and integer programming. John Wiley & Sons, 1998.

[68] SHMOYS, D. and TARDOS, É., "An approximation algorithm for the generalized assignment problem," Mathematical Programming, vol. 62, no. 1-3, pp. 461–474, 1993.

[69] SMITH, S. and BULLO, F., "Target assignment for robotic networks: Asymptotic performance under limited communication," in Proceedings of the American Control Conference, pp. 1155–1160, 2007.

[70] SOLOMON, M., "Algorithms for the vehicle routing and scheduling problems with time window constraints," Operations Research, vol. 35, no. 2, pp. 254 – 265, 1987.

[71] SPANOS, D. and MURRAY, R., "Motion planning with wireless network constraints,"

Proceedings of the American Control Conference, pp. 87 – 92, 2005.

[72] STROUPE, A. W. and BALCH, T., "Value-based action selection for observation with robot teams using probabilistic techniques," Robotics and Autonomous Systems, vol. 50, no. 2, pp. 85–97, 2005.

[73] TAHBAZ-SALEHI, A. and JADBABAIE, A., "On consensus over random networks," in 44th Annual Allerton Conference, Citeseer, 2006.

[74] TOTH, P. and VIGO, D., The vehicle routing problem, vol. 9. Siam, 2002.

[75] TOVEY, C., LAGOUDAKIS, M., JAIN, S., and KOENIG, S., "The generation of bidding rules for auction-based robot coordination," in From Swarms to Intelligent Automata Volume III, pp. 3–14, Springer, 2005.

[76] YAMADA, T. and NASU, Y., "Heuristic and exact algorithms for the simultaneous assignment problem," European Journal of Operational Research, vol. 123, no. 3, pp. 531–542, 2000.

[77] YU, Z., JINHAI, L., GUOCHANG, G., RUBO, Z., and HAIYAN, Y., "An implementation of evolutionary computation for path planning of cooperative mobile robots," in Intelligent Control and Automation, vol. 3, pp. 1798–1802, 2002.

[78] ZAVLANOS, M. and PAPPAS, G., "Dynamic assignment in distributed motion planning with local coordination," IEEE Transactions on Robotics, vol. 24, no. 1, pp. 232–242, 2008.

[79] ZAVLANOS, M., SPESIVTSEV, L., and PAPPAS, G., "A distributed auction algorithm for the assignment problem," in IEEE Conference on Decision and Control, pp. 1212–1217, 2008.