Heterogeneous Multi-Robot Routing

Smriti Chopra and Magnus Egerstedt

***Abstract*— We consider the problem of routing multiple robots to service spatially distributed requests at specified time instants, where each robot, as well as each request, is associated with one or more skills (or functions). A request can be serviced by a robot as long as the robot has *at least* one skill in common with the skill set of that request. We characterize the feasibility aspects of such a heterogeneous routing problem, and provide algorithms for finding the minimum number of robots required to service the requests, and for constructing the corresponding paths of the robots.**

I. Introduction

Multi-robot routing is a well researched topic in robotics, that requires multiple robots to visit a set of spatially distributed requests with routes that optimize certain criteria [1, 2]. In this paper, we consider such a routing problem with an added temporal constraint that associates with each spatial request, a particular time instant at which the request must be serviced. We call such a request a *spatio-temporal* request. Moreover, we introduce heterogeneity into the problem, by associating one or more skills with each request. Additionally, each robot possesses one or more skills, and can service a request as long as it has *at least* one skill in common with the skill set of that request.

Many problems in combinatorial optimization are associated with multi-robot routing, a few examples being the multiple traveling salesman problem [3], and the vehicle routing problem with its many variations [4, 5]. However, such problems are computationally expensive to solve. Moreover, many applications require that spatial requests be serviced in a synchronized and sequenced manner, thus motivating the addition of temporal constraints to such requests. It is shown in [6], that such a *spatio-temporal* construction is convenient for applying the framework of assignment problems towards finding solutions, with the resulting reduction in complexity.

Heterogeneity is considered an important facet of multi-robot routing [7, 8, 9], with applications in several domains like search and rescue, sensor deployment, transportation on demand, and assembly. Heterogeneous routing problems may differ on the basis of different metrics for heterogeneity, like movement speed and task execution speed of robots [10], or on the basis of different multi-robot task allocation models as per the taxonomy in [11], (for instance, a model where each robot can perform at most a single task at a given time, and each task requires multiple robots, is considered in [12]).

Smriti Chopra and Magnus Egerstedt are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30312, USA, Email: smriti.chopra@gatech.edu,magnus@ece.gatech.edu

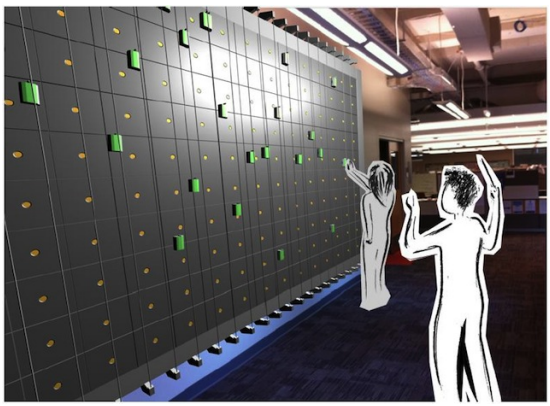

Fig. 1: A rendering of the *Robot Music Wall* concept.

In this paper, 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 (requests are *spatio-temporal* in nature). We formulate a generalized heterogeneous framework that allows robots as well as requests to have associated skill sets, and provide feasibility and minimality results for the corresponding routing problem, as well as an algorithm for constructing the routes of the robots.

A Motivating Example - The Robot Music Wall

Consider a two-dimensional magnetic-based surface (wall) with a grid of strings in different pitches that generate sound when plucked. Distinct positions on the wall correspond to distinct sound frequencies, i.e. distinct notes of an instrument. Multiple robots with the ability to traverse the wall can reach these positions and pluck at the strings above them.

With this set-up, we can interpret any piece of music consisting of a series of notes from multiple instruments, to be played at specified time instants, as a series of corresponding *spatio-temporal* requests (timed positions) on the music wall. We call such a series a *Score*, which contains positions that must be reached at specified time instants[1]. Multiple robots, where each robot can play one or more instrument, are routed to service such timed positions, thereby effectively "playing" the piece of music associated with the timed positions on the wall.

[1]Though this paper solves the generalized version of the routing problem that associates a *set of skills* with each *spatio-temporal* request, for the purpose of musical demonstration, a *single* skill (instrument) is associated with each request (w.l.o.g.).

II. Problem Setup

We let $T = \{t_1, t_2, ..., t_n\}$ denote the set of n discrete time instants over which the *Score* is defined, where $t_1 < ... < t_n$. We let P_i denote the corresponding set of planar positions that require simultaneous servicing at time instant 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.,

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

We let $\mathcal{K}$ be the maximum number of positions that require simultaneous servicing at any time instant in T, i.e.,

$$\mathcal{K} = \max_{i \in \{1,...,n\}} |P_i| \tag{2}$$

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

$$Sc_i = \{(P_{i,\alpha}, t_i) \,|\, \alpha \in \{1, ..., |P_i|\}\} \tag{3}$$

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 for servicing timed positions in the *Score*. Moreover, we let $M_{pos} : Sc \to 2^L \setminus \emptyset$ 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 $(P_{i,\alpha}, t_i) \in Sc$.

For a given set of r robots, denoted by $R = \{1, 2, ..., r\}$, we let $P_0 = \{P_{0,\alpha} \,|\, \alpha \in \{1, ..., r\}\}$ be 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 $M_{rbt} : R \to 2^L \setminus \emptyset$.

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.

III. Feasibility

In this section, we discuss the feasibility aspects of the heterogeneous routing problem, i.e. if it is even possible to execute the *Score* with a given set of resources.

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

$$\bigcup_{p \in R} A(p) = Sc \tag{4}$$

$$A(p) \cap A(q) = \emptyset \quad \forall\, p, q \in R, p \neq q \tag{5}$$

$$(P_{i,\alpha}, t_i), (P_{j,\beta}, t_j) \in A(p) \Rightarrow i \neq j \quad if\, \alpha \neq \beta \tag{6}$$

$$(P_{i,\alpha}, t_i) \in A(p) \Rightarrow M_{pos}((P_{i,\alpha}, t_i)) \cap M_{rbt}(p) \neq \emptyset \tag{7}$$

Equation (4) states that every timed position in the *Score* is assigned, while Equation (5) states that no two robots are assigned the same timed position. Equation (6) states that a robot is not assigned more than one position at a given time instant, and Equation (7) 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.

A. Establishing Feasibility

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 2.

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 1: 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,

$$\bigcup_{s_j \in S_{grp}} \Pi_i(s_j) = Sc_i \tag{8}$$

$$\Pi_i(s_j) \cap \Pi_i(s_k) = \emptyset \quad \forall\, s_j, s_k \in S_{grp}, s_j \neq s_k \tag{9}$$

$$s_j \cap M_{pos}((P_{i,\alpha}, t_i)) \neq \emptyset \quad \forall (P_{i,\alpha}, t_i) \in \Pi_i(s_j) \tag{10}$$

$$|\Pi_i(s_j)| \leq r_j \in R_{rgp} \quad \forall s_j \in S_{grp} \tag{11}$$

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 $\Pi_i : S_{grp} \to 2^{Sc_i}$ such that Equations (8) - (11) 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 (10), we see that each timed position contained in $\Pi_i(s_j)$ has a skill in common with s_j. Moreover, Equation (11) 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 (8) and (9) 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 unique robot in R. Consequently, we can construct a mapping $A : R \to 2^{Sc}$ by combining the above mentioned robot assignments to timed positions, over *all* time instants in the *Score*. Moreover, A satisfies the conditions in Definition 2, 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 $A : R \to 2^{Sc}$ as per Definition 2. For such a mapping A, by definition, there exist Π_is that satisfy Equations (8) - (11). ■

For convenience, we let $\Pi = \{\Pi_i \,|\, i \in \{1, ..., n\}\}$ denote the set of Π_is that satisfy Equations (8) - (11) of Lemma 1.

B. 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 following:

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

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

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

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

Assuming $(Sc, R, L, M_{pst}, M_{rbt})$ is *feasible*, we know from Lemma 1 that at every time $t_i \in T$, there exists a function $\Pi_i : S_{grp} \to 2^{Sc_i}$ such that Equations (8) - (11) 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 $\max_{i\in\{1,...,n\}} r^\Pi_{i,j}$. We denote this number by r^Π_j. For convenience, we let $R^\Pi_{grp} = \{r^\Pi_j \,|\, j \in \mathcal{J}\}$. Finally, the total number of robots required over all time instants *and* all skill sets, denoted by r_Π, is given by,

$$r^\Pi = \sum_{r^\Pi_j \in R^\Pi_{grp}} r^\Pi_j \quad \leq r \tag{12}$$

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 assignment problem that searches for a Π, i.e. a set of Π_is that satisfy Equations (8) - (11) of Lemma 1, thereby ensuring feasibility, while minimizing the total number of robots required, r_Π, given by Equation (12).

In order 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 $l(j, i, \alpha)$, we state the assignment problem as follows,

$$\min_{l} \quad \sum_{j\in\mathcal{J}} \left(\max_{i\in\mathcal{I}} \sum_{\alpha\in\mathcal{A}_i} c(j, i, \alpha)\, l(j, i, \alpha) \right) \tag{13}$$

such that,

$$l(j, i, \alpha) \in \{0, 1\} \tag{14}$$

$$\sum_{j\in\mathcal{J}} c(j, i, \alpha)\, l(j, i, \alpha) = 1 \quad \forall i \in \mathcal{I}, \alpha \in \mathcal{A}_i \tag{15}$$

$$\max_{i\in\mathcal{I}} \sum_{\alpha\in\mathcal{A}_i} c(j, i, \alpha)\, l(j, i, \alpha) \leq r_j \quad \forall j \in \mathcal{J} \tag{16}$$

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 $\Pi_i : S_{grp} \to 2^{Sc_i}$ 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 (15) and (16) ensure that Π_i satisfies Equations (8) - (11) of Lemma 1, for every time $t_i \in T$, thereby ensuring feasibility, while (13) minimizes the total number of robots required.

We proceed to describe the *MinBots* algorithm, that solves the assignment problem defined above.

The MinBots Algorithm

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[2] (set of Π_is that satisfy Equations (8) - (11) of Lemma 1). Given Π, the algorithm calculates the cost to be minimized, i.e. the total number of robots required, r^Π, using Equation (12). 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 $s_{j^\star} \in S_{grp}$[3], the *MinBots*

[2]Equivalent to finding an l that satisfies Equations (14) to (16), since we can construct Π from such an l.

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

Algorithm 1 *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 (8) - (11) of Lemma 1 {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 (13)-(16)}

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 $\mathcal{I}' = \{i \in \mathcal{I} \,|\, r^{\Pi}_{i,j^\star} = r^{\Pi}_{j^\star}\}$ 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 in order to maintain feasibility, $\hat{\Pi}_i$ must satisfy Equations (8) - (11) of Lemma 1.

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[4]:

$$\min_{l} \quad \sum_{\alpha \in \mathcal{A}_i} c(j^\star, \alpha)\, l(j^\star, \alpha) \tag{17}$$

such that,

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

$$\sum_{j \in \mathcal{J}} c(j, \alpha)\, l(j, \alpha) = 1 \quad \forall \alpha \in \mathcal{A}_i \tag{19}$$

$$\sum_{\alpha \in \mathcal{A}_i} c(j, \alpha)\, l(j, \alpha) \leq r^{\Pi}_j \quad \forall j \in \mathcal{J} \tag{20}$$

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 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 2: 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 (20) 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 1: Given a quintuple $(Sc, R, L, M_{pst}, M_{rbt})$ that is *feasible*, 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 $(Sc, R', L, M_{pst}, M_{rbt})$ is *feasible*
(b) there exists no $R^\star \subset R$, $|R^\star| < r^\star$, such that $(Sc, R^\star, L, M_{pst}, M_{rbt})$ is *feasible*

[4]The assignment problem is feasible, since Π_i satisfies Equations (8) - (11) of Lemma 1.

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 (8) - (11) of Lemma 1. Hence, using Lemma 1, we can conclude that there exists some $R' \subseteq R$, $|R'| = r^{\star}$, such that $(Sc, R', L, M_{pst}, M_{rbt})$ is *feasible* or in other words, condition (a) of Theorem 1 is satisfied.

For condition (b) of Theorem 1, 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 $(Sc, R, L, M_{pst}, M_{rbt})$ 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 2, 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 $(Sc, R^{\star}, L, M_{pst}, M_{rbt})$ is *feasible*, i.e. condition (b) of Theorem 1 is satisfied. ∎

Computational Complexity: The assignment problem described in the *ReduceCost* sub-algorithm can be solved using the *Hungarian Method* [13], the fastest version of which has complexity $\mathcal{O}(N^3)$, for N stages [14]. In the case of the *ReduceCost* sub-algorithm, N equals the total number of robots required, i.e. r^{Π}, where $r^{\Pi} \leq r$. Thus, in order to characterize the complexity of the *MinBots* algorithm, we bound the number of *available* robots per skill set, $r_j \in R_{grp}$, to be no more that $\mathcal{K}$[5], where $\mathcal{K}$ denotes the maximum number of timed positions that require simultaneous servicing. In other words, 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 $r \leq |S_{grp}|\mathcal{K}$.

IV. Path Generation

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*.

The PathGen algorithm

We proceed to explain the *PathGen* algorithm, and the *Assign* sub-algorithm it utilizes.

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

Algorithm 2 *PathGen* $(Sc, R, L, P_0, M_{pos}, M_{rbt})$

1: Define $A : R \rightarrow 2^{Sc}$, and initialize as follows: $A(p) = \emptyset \quad \forall p \in R$
2: Define $P_{cur} : R \rightarrow \mathbb{R}^2$, 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: $H^{\star} \leftarrow$ *Assign* $(Sc_i, R, P_{cur}, M_{pos}, M_{rbt})$ {Find $H^{\star} : R' \rightarrow Sc_i$, $R' \subseteq R$, that encodes the new positions occupied by all robots in R'}
5: Using $H^{\star} : R' \rightarrow Sc_i$, $R' \subseteq R$, 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

Assign $(Sc_i, R, P_{cur}, M_{pos}, M_{rbt})$ *:* 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. The sub-algorithm finds some $R' \subseteq R$ such that firstly, the restricted function $H|_{R'}R' \rightarrow Sc_i$ 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 [15] described as follows,

$$\min_{l} \sum_{p \in R} \sum_{\alpha \in \mathcal{A}_i} ||P_{i,\alpha} - P_{cur}(p)||\, l(p, \alpha) \tag{21}$$

subject to:

$$l(p, \alpha) \in \{0, 1\} \tag{22}$$

$$\sum_{p \in R} l(p, \alpha) = 1, \quad \forall\, \alpha \in \mathcal{A}_i \tag{23}$$

$$\sum_{\alpha \in \mathcal{A}_i} l(p, \alpha) \leq 1, \quad \forall\, p \in R \tag{24}$$

$$l(p, \alpha) = 1 \Rightarrow M_{rbt}(p) \cap M_{pos}((P_{i,\alpha}, t_i)) \neq \emptyset \tag{25}$$

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 (4) - (7) of Definition 2. 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 $r \leq |Sc|$.

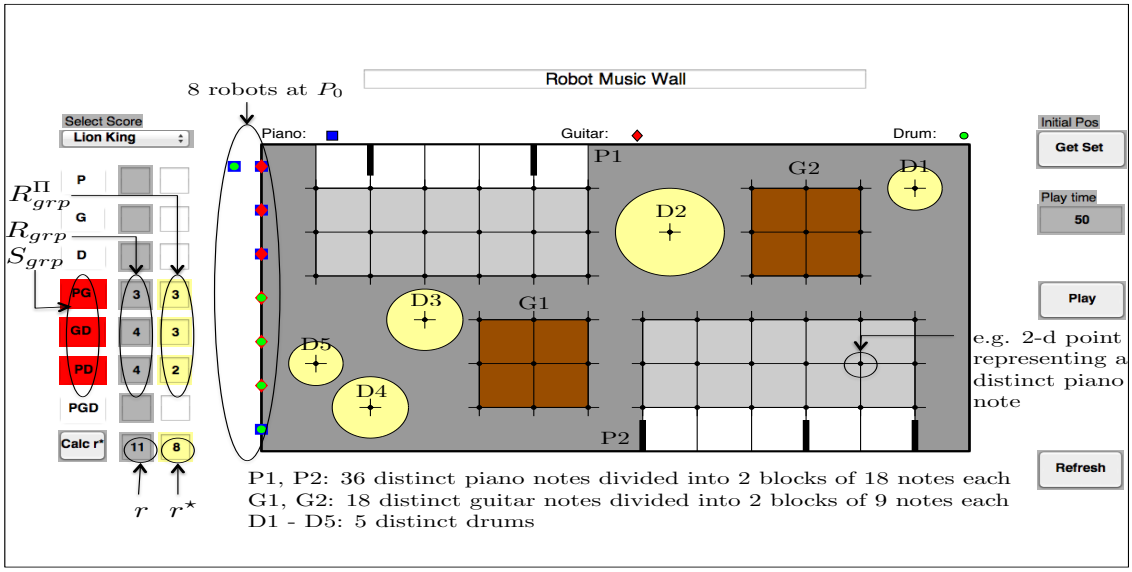

Fig. 2: A simulated GUI of an example of the *Robot Music Wall*, with coordinates representing either piano notes, guitar notes or drums. For a user specified choice of $S_{grp} = \{\{p, g\}, \{g, d\}, \{p, d\}\}$ and $R_{grp} = \{3, 4, 4\}$, *MinBots* provides $r^\star = 8$, and $R^{\Pi}_{grp} = \{3, 3, 2\}$, 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.

V. Simulations

To implement the heterogeneous routing problem central to this paper, we simulated an example of the *Robot Music Wall* in MATLAB (Figure 2), and created the *Score* associated with the song "Can You Feel the Love Tonight" by Elton John (performed in The Lion King). For a user specified set of resources, (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*, and a corresponding distribution of robots per skill set, R^{Π}_{grp} (need not be unique), using the *MinBots* algorithm. Moreover, we constructed the corresponding paths of the robots, using the *PathGen* algorithm[6].

References

[1] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376 – 378, 2005.

[2] A. Mosteo, L. Montano, and M. Lagoudakis, "Multi-robot routing under limited communication range," *IEEE International Conference on Robotics and Automation*, pp. 1531 – 1536, 2008.

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

[4] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482 –1504, 2011.

[5] L. Bodin, B. Golden, A. Assad, and M. Ball, "Routing and scheduling of vehicles and crews: The state of the art." *Computers and Operations Research*, vol. 10, no. 2, pp. 63–211, 1983.

[6] S. Chopra and M. Egerstedt, "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.

[7] A. Howard, L. Parker, and G. Sukhatme, "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.

[8] L. Parker, "Adaptive heterogeneous multi-robot teams," *Neurocomputing*, vol. 28, pp. 75 – 92, 1999.

[9] R. Baldacci, M. Battarra, and D. Vigo, *Routing a heterogeneous fleet of vehicles*. Springer, 2008.

[10] M. Franceschelli, D. Rosa, C. Seatzu, and F. Bullo, "Gossip algorithms for heterogeneous multi-vehicle routing problems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, pp. 156 – 174, 2013.

[11] B. Gerkey and M. Matarić, "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.

[12] M. Koes, I. Nourbakhsh, and K. Sycara, "Heterogeneous multirobot coordination with spatial and temporal constraints," *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pp. 1292–1297, 2005.

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

[14] E. Lawler, *Combinatorial optimization: Networks and matroids*. Holt, Rinehart and Winston, New York, 1976.

[15] U. Derigs, "The shortest augmenting path method for solving assignment problems - motivation and computational experience," *Annals of Operations Research*, vol. 4, pp. 57–102, 1985.

[6]Although immaterial to the underlying theory, we chose to move robots between assigned timed positions in straight line trajectories, with constant velocities that ensured their timely arrival.