Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

A Distributed Version of the Hungarian Method for Multirobot Assignment

(Article begins on next page)

26 April 2024

A Distributed Version of the Hungarian Method for Multi-Robot Assignment

Smriti Chopra, Giuseppe Notarstefano, Matthew Rice, and Magnus Egerstedt

Abstract—In this paper, we propose a distributed version of the Hungarian Method to solve the well known assignment problem. In the context of multi-robot applications, all robots cooperatively compute a common assignment that optimizes a given global criterion (e.g. the total distance traveled) within a finite set of local computations and communications over a peer-to-peer network. As a motivating application, we consider a class of multi-robot routing problems with "spatio-temporal" constraints, i.e. spatial targets that require servicing at particular time instants. As a means of demonstrating the theory developed in this paper, the robots cooperatively find online, suboptimal routes by applying an iterative version of the proposed algorithm, in a distributed and dynamic setting. As a concrete experimental test-bed, we provide an interactive "multi-robot orchestral" framework in which a team of robots cooperatively plays a piece of music on a so-called orchestral floor.

NOMENCLATURE

In the following table, we present the nomenclature used in this paper.

Table of Notation

V	$\triangleq$	Vertex partitioning of two disjoint sets R (robots) and P (targets) respectively, denoted by $V = (R, P)$
E_{orig}^i	$\triangleq$	Edge set containing the edges between a robot $i \in R$ and every target in P
w_{orig}^i	$\triangleq$	Weight function corresponding to the edge set E_{orig}^i
G_{orig}^i	$\triangleq$	Robot i's original information, denoted by the bipartite weighted graph $G_{orig}^i = (V, E_{orig}^i, w_{orig}^i)$
E_y^i	$\triangleq$	Robot i's equality subgraph edges
E_{cand}^i	$\triangleq$	Robot i's candidate edges
E_{lean}^i	$\triangleq$	Edge partitioning of two disjoint sets E_y^i and E_{cand}^i respectively, denoted by $E_{lean}^i = (E_y^i, E_{cand}^i)$
w_{lean}^i	$\triangleq$	Weight function corresponding to the edge set E_{lean}^i
G_{lean}^i	$\triangleq$	Robot i's information, denoted by the bipartite weighted graph $G_{lean}^i = (V, E_{lean}^i, w_{lean}^i)$
y^i	$\triangleq$	Robot i's vertex labeling function
γ^i	$\triangleq$	Robot i's counter value
$\mathbb{G}^i$	$\triangleq$	Robot i's full state, given by $\mathbb{G}^i = (G_{lean}^i, y^i, \gamma^i)$
M^i	$\triangleq$	Robot i's Maximum Cardinality Matching
V_c^i	$\triangleq$	Robot i's MInimum Vertex Cover

Smriti Chopra, Matthew Rice and Magnus Egerstedt are with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA; Emails: smriti.chopra@gatech.edu, mrice32@gatech.edu, magnus@ece.gatech.edu. Giuseppe Notarstefano is with the Department of Engineering, Universit del Salento, Lecce, Italy; Email: giuseppe.notarstefano@unisalento.it

I. INTRODUCTION

Assignment problems are an integral part of combinatorial optimization, with wide applicability in theory as well as practice [1, 2, 3, 4]. Various techniques have been proposed for solving such problems (see [5, 6, 7] for early references). For scenarios involving multiple mobile robots, assignment problems often comprise of finding a one-to-one matching between robots and tasks, while minimizing some assignment benefit. Moreover, a frequent requirement is the need for a distributed framework, since an infrastructure that supports a centralized authority is often not a feasible option (prohibitively high cost for global computation and information). It is preferable that robots coordinate with one another to allocate and execute individual tasks, through an efficient, distributed mechanism - a feat often challenging due to the limited communication capabilities and global knowledge of each robot.

In this paper, we address such assignment problems with *linear* objective functions, formally called Linear Sum Assignment Problems (LSAPs) [8], under a distributed setting in which robots communicate locally with "adjacent neighbors" via a dynamic, directed information-exchange network. Among centralized algorithms, the Hungarian Method [9] was the first to compute an optimal solution to the LSAP in finite time, and as such, forms the basis of our proposed distributed algorithm.

In cooperative robotics, assignment problems often form building blocks for more complex tasks, and have been widely investigated in the literature [10, 11, 12, 13]. In particular, auction-based (market) algorithms are a very popular approach towards task assignments (see [14] for a survey). Such algorithms require robots to bid on tasks, rendering them more or less attractive based on the corresponding prices computed [15, 16]. A generic framework and a variety of bidding rules for auction-based multi-robot routing have been proposed in [17]. An auction algorithm for dynamic allocation of tasks to robots in the presence of uncertainties and malfunctions has been proposed and tested in [18]. Auction algorithms, though computationally efficient, usually require a coordinator or shared memory. In [19], the authors develop an auction algorithm without such constraints, and apply it towards multi-robot coordination in [20]. In particular, the agents obtain updated prices, required for accurate bidding, in a multi-hop fashion using only local information. The authors prove that the algorithm converges to an assignment that maximizes the total assignment benefit within a linear approximation of the optimal one, in $\mathcal{O}(\Delta n^3 \left\lceil \frac{\max\{c_{i,j}\} - \min\{c_{i,j}\}}{\epsilon} \right\rceil)$ iterations.

In [21], a market-based decision strategy is proposed for decentralized task selection, and a consensus routine, based on local communications, is used for conflict resolutions and agreement on the winning bid values. In [22], an auction algorithm is proposed to provide an almost optimal solution to the assignment problem with set precedence constraints. The algorithm is first presented in a shared-memory scenario, and later extended via a consensus algorithm, to a distributed one.

Game-theoretic formulations for solving vehicle-target assignment problems are discussed [30, 31], where robots are viewed as self-interested decision makers, and the objective is to optimize a global utility function through robots that make individually rational decisions to optimize their own utility functions. Among other decentralized techniques, coordination algorithms for task allocation that use only local sensing and no direct communication between robots have been proposed in [23]. Additionally, consensus based approaches that typically require the robots to converge on a consistent situational awareness before performing the assignment have been explored in [24, 25, 26]. 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 [27, 28], though they are computationally expensive, especially in comparison to more streamlined algorithms, developed for the purpose of solving assignment problems.

In [29, 35], the authors propose a distributed version of the Hungarian Method, similar to the contribution in this paper. They show that their algorithm converges in $\mathcal{O}(n^3)$ cumulative time, with $\mathcal{O}(n^3)$ number of messages exchanged among the robots, and no coordinator or shared memory. In particular, their algorithm involves root robots that (i) initiate message exchange with other robots in the network via a depth-first search (DFS), and (ii) synchronize the decision rounds (iterations, each containing multiple communication hops) across all robots.

The main distinctive feature of this paper is the redesign of the popular (centralized) Hungarian Method, under a distributed computation model, characteristic of traditional multi-robot applications - where every iteration of an algorithm ideally involves multiple anonymous agents performing two tasks: 1) exchanging information with their neighbors (via single-hop communication), and 2) executing identical computation routines. Our primary objective is not to improve convergence speeds, or information overheads of existing methods, but to remain comparable while providing a novel, distributed implementation of the centralized method. We prove that our algorithm converges in $\mathcal{O}(n^3)$ iterations, and show through simulation experiments with varying problem sizes, that the average convergence is much faster in practice, thus making our algorithm relevant to current literature.

The contribution of this paper is twofold: (i) As the main contribution, we develop a distributed version of the Hungarian Method to enable a team of robots to cooperatively compute optimal solutions to task assignment problems (LSAPs), without any coordinator or shared memory. Specifically, each robot runs a local routine to execute ad-hoc sub-steps of the centralized Hungarian Method, and exchanges estimates of the solution with neighboring robots. We show that in finite time (or in a finite number of communication rounds $\mathcal{O}(r^3)$ if executing synchronously, with r being the total number of robots in the system), all robots converge to a *common* optimal assignment (the LSAP can have multiple optimal solutions). Through simulation experiments over varying problem sizes, we characterize the average number of iterations required for convergence, as well as the computational load per robot. (ii) We demonstrate our proposed algorithm by extending it towards a class of "spatio-temporal" multi-robot routing problems previously introduced in [36, 37], now considered under a distributed and dynamic setting. In essence, the robots find online, sub-optimal routes by solving a sequence of assignment problems iteratively, using the distributed algorithm for each instance. As a motivating application and concrete experimental test-bed, we develop the "multi-robot orchestral" framework, where spatio-temporal routing is musically interpreted as "playing a series of notes at particular time instants" on a so-called orchestral floor (a music surface where planar positions correspond to distinct notes of different instruments). Moreover, we allow a user to act akin to a "conductor", modifying the music that the robots are playing in real time through a tablet interface. Under such a framework, we demonstrate the theory developed in this paper through simulations and hardware experiments.

The remainder of this paper is organized as follows: In Section II, we briefly review the assignment problem, and the Hungarian Method used for solving it. In Section III, we set-up the distributed version of the assignment problem central to this paper, while in Section IV, we provide a description of our proposed algorithm. We discuss convergence and optimality in Section V, followed by the motivating application of spatio-temporal multi-robot routing in Section VI.

II. A REVIEW OF THE LINEAR SUM ASSIGNMENT PROBLEM AND THE HUNGARIAN METHOD

In this section, we consider the Linear Sum Assignment Problem (LSAP) under a centralized setting [8], before we delve into its proposed distributed counterpart. We revisit 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 R and P respectively, such that no two vertices in the same set are adjacent. In general, we say that the graph G has bipartition (R, P).
- **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 [38]). In 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 [39]), can be extended to finding a corresponding minimum vertex cover $V_c \subset V$. □

A. The Linear Sum Assignment Problem

Using the definitions presented above, we proceed to review the formal, graph theoretic interpretation of the LSAP.

Minimum Weight Bipartite Matching Problem (P)

"Given a graph $G = (V, E)$ with bipartition (R, P) 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*[1], i.e. there exists an edge between every vertex $i \in R$, and every vertex $j \in P$, and *balanced*[2], i.e. $|R| = |P| = |V|/2$. Hence, a maximum cardinality matching M is always a perfect matching, i.e. $|M| = |V|/2$. Next, we review the **dual** of the above problem:

Dual of Minimum Weight Bipartite Matching Problem (D)

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

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

$$E_y = \{(i, j) \mid y(i) + y(j) = w(i, j)\} \tag{1}$$

and the **slack** of an edge (i, j) is defined as,

$$\text{slack}(w, y, i, j) = w(i, j) - (y(i) + y(j)) \tag{2}$$

B. The Hungarian Method

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 [9], the first primal-dual algorithm developed for solving the LSAP.

Theorem 1 (Kuhn-Munkres): Given a bipartite graph $G = (V, E)$ with bipartition (R, P), a weight function $w : E \to \mathbb{R}_{\geq 0}$, 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 $M \subseteq E_y$,

[1]by adding edges with prohibitively large weights denoted by $\mathfrak{M}$.
[2]by adding dummy vertices and associated 0-weight edges.

i.e. each edge in M is also in the set of equality subgraph edges E_y, given by (1). □

From this point onwards, for notational convenience, we will denote the *weighted, bipartite graph* by $G = (V, E, w)$, i.e. a tuple consisting of the vertex set V, the edge set E and the corresponding edge weight function w.

We proceed to provide a brief description of the Hungarian Method[3] that will assist us in explaining our proposed distributed algorithm in later sections of this paper (see Figures 1 and 2 for corresponding instances).

---

function Hungarian_Method (G)

% Initialization Step

y = arbitrary feasible labeling, example:
$y(i \in R) = \min_{j \in P} w(i, j)$ and $y(j \in P) = 0$
E_y = equality subgraph edges using (1)
(M, V_c) = maximum cardinality matching and corresponding minimum vertex cover, given (V, E_y) (see Remark 1)

while M is not a perfect matching **do**

% Step 1(a)

  for $i \in R \setminus R_c$ **do**
    Choose any $j^\star \in \arg\min_{j \in P \setminus P_c} \text{slack}(w, y, i, j)$, and set $e^i_{cand} = (i, j^\star)$ using (2)
  end for
  $E_{cand} = \cup_{i \in R \setminus R_c} \{e^i_{cand}\}$

% Step 1(b)

  $\delta = \min_{(i,j) \in E_{cand}} \text{slack}(w, y, i, j)$
  $y(i) = y(i) - \delta, \ \forall i \in R_c$
  $y(j) = y(j) + \delta, \ \forall j \in P \setminus P_c$

% Step 2

  E_y = equality subgraph edges
  (M, V_c) = maximum cardinality matching and corresponding minimum vertex cover, given (V, E_y)

end while

Remark 2: As mentioned in Step 1, the *selection* of the candidate edges is done based on the minimum vertex cover $V_c = (R_c, P_c)$. In particular, the set of candidate edges E_{cand} represents the edges between vertices in $R \setminus R_c$ and vertices in $P \setminus P_c$, i.e. edges between the so-called *uncovered* vertices in R and *uncovered* vertices in P (see Figure 2a for an example). □

Without delving into details, we provide an auxiliary lemma, followed by a quick proof sketch that shows the Hungarian Method converges to an optimal solution (see [4, 40] for details). We will rely on these fundamental results in later sections of the paper, where we discuss the convergence properties of our proposed distributed algorithm.

[3]There are more ways than one to implement the primal-dual Hungarian Method - we describe the implementation that forms the basis of our proposed algorithm.

(a) An example of a weighted, bipartite graph $G = (V, E, w)$.

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

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

(d) Given E_y, a maximum cardinality matching M (red edges), and a corresponding minimum vertex cover $V_c = (R_c, P_c)$ (red vertices).
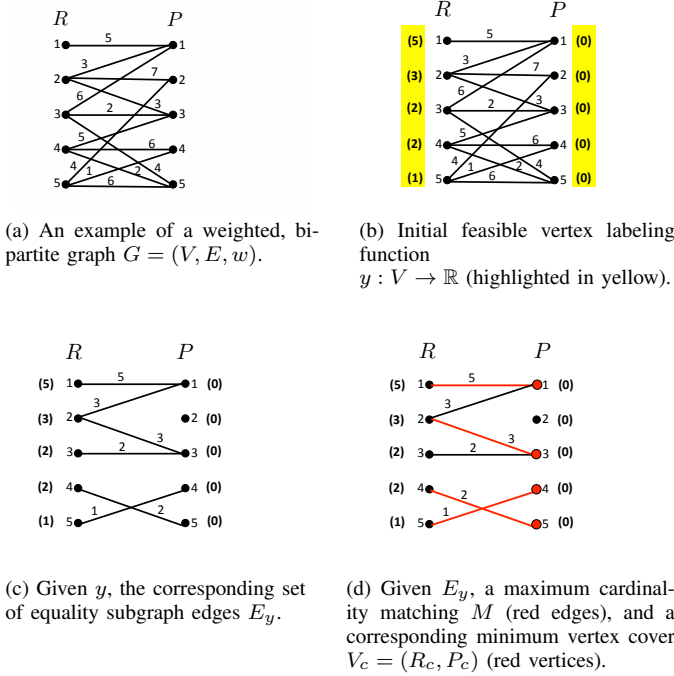
Fig. 1: An instance of the Initialization step of the Hungarian_Method.

Lemma 1: Given a weighted, bipartite graph $G = (V, E, w)$, with bipartition (R, P), a feasible vertex labeling function y, and a corresponding maximal matching M, every two-step iteration (Step 1 and Step 2) of the Hungarian_Method results in the following: (i) an updated y that remains feasible, and (ii) an increase in the matching size $|M|$, or no change in the matching M, but an increase in $|R_c|$ (and corresponding decrease in $|P_c|$, such that $|R_c| + |P_c| = |M|$).[4] □

Remark 3 (Proof sketch of the Hungarian_Method*):* The above stated Lemma 1 ensures that the size of a matching M increases after a finite number of two-step iterations (worst-case r, where $r = |R| = |V|/2$). Since the algorithm converges when M is perfect, i.e. $|M| = r$, Lemma 1 in conjunction with Theorem 1 proves that the Hungarian_Method converges to an optimal solution (perfect matching with minimum cost), after $\mathcal{O}(r^2)$ two-step iterations. Each two-step iteration requires $\mathcal{O}(r^2)$ time, yielding a total running time of $\mathcal{O}(r^4)$ (through certain modifications, this running time can be reduced to $\mathcal{O}(r^3)$). □

Now that we have reviewed the LSAP, as well as the Hungarian Method used for solving it, we proceed to setup the distributed problem central to this paper.

III. DISTRIBUTED PROBLEM SETUP

Similar to the previous section, let $R = \{1, 2, ..., r\}$ denote a set of r robots, and $P = \{1, 2, ..., p\}$ denote a set of p targets,

[4]Either $|R_c|$ increases and $|P_c|$ decreases, or $|P_c|$ increases and $|R_c|$ decreases, depending on the particular implementation of the algorithm employed for finding M and V_c.

(a) Step 1(a): For the given minimum vertex cover V_c, the isolated set of candidate edges (green edges). Step 1(b): The edge with minimum slack (δ) is identified.

(b) Step 1(b) contd.: 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 (red edges), and a corresponding minimum vertex cover $V_c = (R_c, P_c)$ (red vertices).
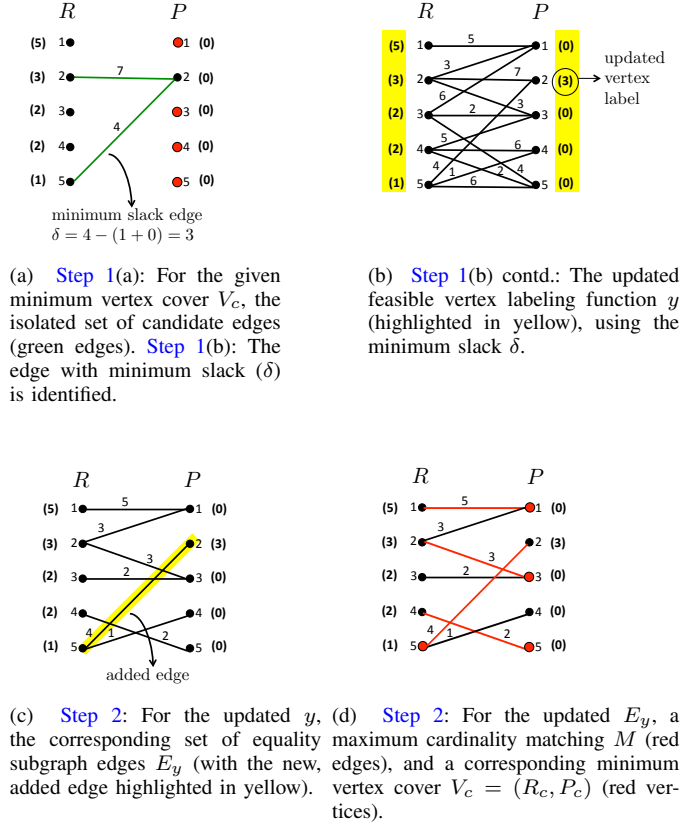
Fig. 2: An instance of a two-step iteration (Step 1 and Step 2) of the Hungarian_Method.

where $r \geq p$. Let $P^i \subseteq P$ be the set of targets that robot $i \in R$ can be assigned to, with the associated cost function $c^i : P^i \to \mathbb{R}$. 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.

Remark 4: We can generate the problem data for the corresponding centralized assignment problem (Section II), as the weighted bipartite graph $G = (V, E, w)$, where,

- The edge set E is given by $E = \{(i, j) \,|\, j \in P^i\} \; \forall i \in R$,
- The weight function $w : E \to \mathbb{R}$ is given by $w((i, j)) = c^i(j), \forall (i, j) \in E$.

As mentioned before, we can modify G to ensure it is 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 $c^\star = c(M) = \sum_{e \in M} w(e)$.

Communication network: We model the communication between the robots by a time-varying directed graph $\mathcal{G}_c(t) =$

$(R, E_c(t))$, $t \in \mathbb{R}_{\geq 0}$. In such a graph, an edge from robot i to robot j at some time t implies that robot i can communicate with robot j at that time instant. 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 1: For every time instant $t \in \mathbb{R}_{\geq 0}$, the directed graph $\mathcal{G}_c(t)$ is strongly connected, i.e., there exists a directed path from every robot, to every other robot in $\mathcal{G}_c(t)$. $\square$

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 1.

Before proceeding to the algorithm central to this paper, we would like to make three remarks on the proposed set-up.

- We have introduced two graphs which have completely different roles. The first (fixed, wighted, bipartite) graph $G = (V, E, w)$ models the assignment problem (by relating the set of robots R to the set of targets P). The second graph $\mathcal{G}_c(t)$ describes the peer-to-peer communication network among the robots.
- Assumption 1 can be relaxed by requiring the communication graph to be only *jointly* strongly connected over some time period[5]. Note that the relaxation to a jointly strongly connected communication network ties neatly into the framework of an *asynchronous* implementation of any distributed algorithm (e.g. if a robot is still computing, it would have no edges in the underlying communication graph for that time duration, allowing every robot to communicate and compute at its own speed, without any synchronization within the network). However, in order to not overweight the proofs, we prefer to stay with the more stringent condition (strongly connected at *all* times), and assume our algorithm runs synchronously (explained later in detail). We briefly discuss our proposed algorithm in an asynchronous setting in Remark 8, following the synchronous convergence analysis in Section V.
- Degeneracy of assignment problems[6] is of particular concern in a distributed framework, since all robots must converge not only to an optimal solution, 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 1. 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 a common assignment, $\hat{M}$, that is optimal to the centralized

assignment problem, i.e., $\hat{M} \in \mathcal{M}$.

IV. A DISTRIBUTED VERSION OF THE HUNGARIAN METHOD

Drawing from the description of the Hungarian_Method (II-B), we propose the *Distributed-Hungarian* algorithm for solving the Distributed Assignment Problem (III), where $\mathbb{G}^i$ denotes the state of robot i. Specifically, $\mathbb{G}^i$ contains the following three objects:

(i) $G^i_{lean} = (V, E^i_{lean}, w^i_{lean})$: A bipartite graph with vertex partitioning $V = (R, P)$, two disjoint sets of edges E^i_y and E^i_{cand}, denoted by the edge partitioning $E^i_{lean} = (E^i_y, E^i_{cand})$, and a corresponding edge weight function w^i_{lean}[7].
(ii) y^i: A vertex labeling function for G^i_{lean}
(iii) $\gamma^i \in \mathbb{Z}$: A counter variable

Let $T_s = \{t_0, t_1, t_2, \dots\}$ be the set of discrete time instants over which the robots *synchronously* run the algorithm. In other words, at every time instant $t \in T_s$, each robot performs the following two actions repeatedly: (i) it sends a message $msg^i = \mathbb{G}^i$ to each of its outgoing neighbors, and (ii) upon receiving messages from its incoming neighbors, it computes its new state $\mathbb{G}^i$. In this manner, each time instant represents an "iteration" or "communication-round" of the *Distributed-Hungarian* algorithm.

To provide more context, recall that in the Distributed Assignment Problem (III), each robot i has access to (R, P, c^i). Thus, before beginning the algorithm, every robot creates its so-called "original information" in the form of a weighted, bipartite graph $G^i_{orig} = (V, E^i_{orig}, w^i_{orig})$, where,

- The edge set E^i_{orig} is given by $E^i_{orig} = \{(i, j) \,|\, j \in P\}$,
- The weight function $w^i_{orig} : E^i_{orig} \to \mathbb{R}$ is given by,

$$w^i_{orig}((i, j)) = \begin{cases} c^i(j) & j \in P^i, \\ \mathfrak{M} & j \in P \setminus P^i \end{cases}$$

The algorithm is then initialized as follows: At $t = t_0$, each robot i selects an edge $(i, j^\star)$ with minimum weight from its original information G^i_{orig}. Using this edge, the robot initializes its state $\mathbb{G}^i = (G^i_{lean}, y^i, \gamma^i)$, where,

$$G^i_{lean} = (V, (\{(i, j^\star)\}, \emptyset), w^i_{orig}(i, j^\star)) \tag{3}$$

$$y^i(i) = w^i_{orig}((i, j^\star)); \ y^i(j) = 0, \forall j \in \{\{R \setminus \{i\}\} \cup P\} \tag{4}$$

$$\gamma^i = -1 \tag{5}$$

See Figure 3 for an example of the above.

Upon receiving the messages (states) of all incoming neighbors, robot i performs the following steps:

- it calls the Build_Latest_Graph function on all the states in its memory, i.e. $\{\mathbb{G}^i\} \cup (\cup_{j \in \mathcal{N}_I(i,t)}\{\mathbb{G}^j\})$, to obtain a temporary, most-updated state $\mathbb{G}_{tmp}$.
- using $\mathbb{G}_{tmp}$ and its original information G^i_{orig}, robot i calls the Local_Hungarian function to compute its new state $\mathbb{G}^i$.

---

[5]There exists a positive and bounded duration T_c, such that for every time instant $t \in \mathbb{R}_{\geq 0}$, the directed graph $\mathcal{G}_c^{t+T_c}(t) := \bigcup_{\tau=t}^{t+T_c} \mathcal{G}_c(\tau)$

[6]An assignment problem is degenerate when there exists more than one assignment with minimum cost.

[7]The subscript *lean* refers to the sparseness of the graph, with significantly less number of edges than a complete graph.
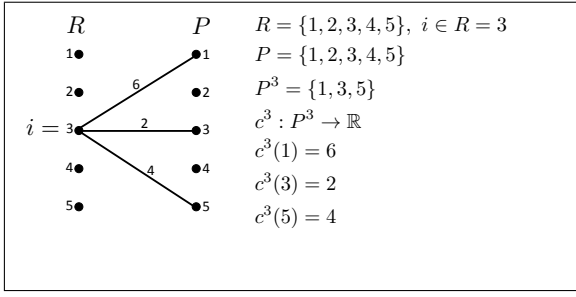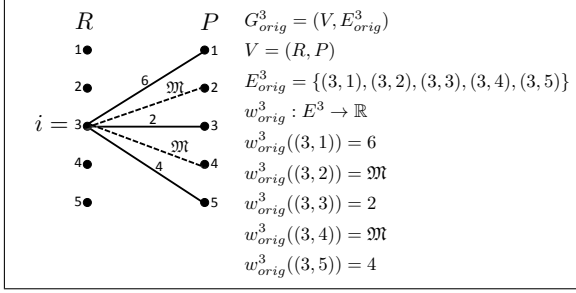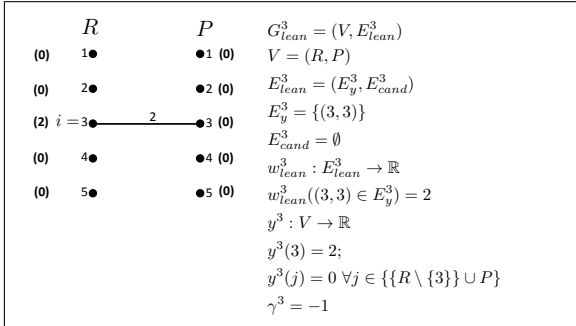
(a) Robot 3's original information (R, P, c^3).

(b) Robot 3's original information in the form of the weighted, bipartite graph graph $G^3_{orig} = (V, E^3_{orig}, w^3_{orig})$.

(c) Robot 3's initial state (at $t = t_0$), $\mathbb{G}^3 = (G^3_{lean}, y^3, \gamma^3)$.

Fig. 3: With the weighted, bipartite graph from Figure 1a as the centralized graph (Remark 4), this figure depicts a single robot's (robot 3) original information, and corresponding initial state, in the *Distributed-Hungarian* algorithm.

We proceed to formally state the Build_Latest_Graph and Local_Hungarian functions.

A. Build_Latest_Graph

Given a set of robots $R' \in R$, and a set of corresponding states $S = \cup_{j \in R'} \{\mathbb{G}^j\}$, the Build_Latest_Graph function returns a resultant, most-updated state $\mathbb{G}_{tmp} = (G_{lean}, y, \gamma)$ that contains the information of *only* those robots that have the highest counter value. We denote such a subset of robots by R_{lead}. If the highest counter value is positive, the function chooses any one robot $j^\star$ in R_{lead}, and sets γ, y and E_y equal to $j^\star$'s corresponding information. However, the function combines the candidate edges of *all* robots in R_{lead}, i.e. $E_{cand} = \bigcup_{j \in R_{lead}} E^j_{cand}$, and sets $G_{lean} = (V, (E_y, E_{cand}), w_{lean})$, where w_{lean} is the corresponding edge weight function.

A special instance of the function occurs when all counter

---

function Build_Latest_Graph (R', S)

$\quad$ **if** (all $\gamma^j = -1$) **then**
$\qquad \gamma = -1$
$\qquad E_y = \bigcup_{j \in R'} E^j_y$
$\qquad E_{cand} = \emptyset$
$\qquad G_{lean} = (V, (E_y, E_{cand}), w_{lean})$
$\qquad y(j) = y^j(j), \forall j \in R'; y(j) = 0, \forall j \in \{\{R \setminus R'\} \cup P\}$

$\qquad$ **if** $|E_y| = |R|$ **then**
$\qquad\quad \gamma = 0$
$\qquad$ **end if**
$\quad$ **else**
$\qquad R_{lead} = \arg\max_{j \in R'} \gamma^j$
$\qquad$ choose any $j \in R_{lead}$ and set $\gamma = \gamma^j; y = y^j; E_y = E^j_y$
$\qquad E_{cand} = \bigcup_{j \in R_{lead}} E^j_{cand}$
$\qquad G_{lean} = (V, (E_y, E_{cand}), w_{lean})$
$\quad$ **end if**
$\quad$ **return** $\mathbb{G} = (G_{lean}, y, \gamma)$

---

values are -1. In this case, the function simply sets $\gamma = -1$, $E_{cand} = \emptyset$, and combines the equality subgraph edges of all robots in R', i.e. $E_y = \bigcup_{j \in R'} E^j_y$. With $G_{lean} = (V, (E_y, E_{cand}), w_{lean})$, the function sets the vertex labels $y(i)$ of every robot i that has an edge in E_y, to the corresponding weight of that edge. If *all* robots in R have an edge in E_y, the function sets γ to 0.

Remark 5: To provide context to the Build_Latest_Graph with respect to the centralized Hungarian_Method, we note here that robot i's most updated state $\mathbb{G}_{tmp}$ contains y, a *globally* feasible vertex labeling function with respect to the centralized graph $G = (V, E, w)$. In other words, robot i's information is a sparse (lean) version of the centralized method's information at the beginning of every two-step iteration. We prove this fact in later sections of the paper (Lemma 2 and extensions). $\square$

B. Local_Hungarian

Given robot i's temporary state $\mathbb{G}_{tmp} = (G_{lean}, y, \gamma)$, and its original information $G^i_{orig} = (V, E^i_{orig}, w^i_{orig})$, the Local_Hungarian function computes robot i's new state $\mathbb{G}^i$ as follows:

For the bipartite graph $G_{lean} = (V, (E_y, E_{cand}), w_{lean})$, and the vertex labeling function y, the Local_Hungarian function computes the maximum cardinality matching M, and the corresponding minimum vertex cover V_c, as per Remark 1. If M is not a perfect matching, the function chooses a single candidate edge from robot i's original information using the Get_Best_Edge sub-function (formally stated below), and adds it to E_{cand}. Note that robot i can contribute a candidate edge *only* if it is uncovered.

Up until this point, the Local_Hungarian function mimics the Hungarian_Method (II-B). However, following this, the

function Get_Best_Edge (G_{orig}^i, y, V_c)

1: **if** $i \in R \setminus R_c$ **then**
2:    Choose any $j^\star \in \arg\min_{j \in P \setminus P_c} \text{slack}(w_{orig}^i, y, i, j)$, and set $e_{cand} = (i, j^\star)$
3: **end if**
4: **return** e_{cand}

function can have one of two outcomes:

- *There exists an uncovered robot with no edge in E_{cand}*
  The above case corresponds to an *incomplete* Step 1(a) of the Hungarian_Method, and as such, the Local_Hungarian function simply sets robot i's new state $\mathbb{G}^i = (G_{lean}, y, \gamma)$, where G_{lean} contains the (possibly) updated E_{cand}.

- *Every uncovered robot has an edge in E_{cand}*
  Such a case corresponds to a *completed* Step 1(a) of the Hungarian_Method, and thus, the Local_Hungarian function continues with Step 1(b) and Step 2, updating y, E_y, M and V_c, and incrementing the counter value γ by 1. Using the updated information, the function resets E_{cand} to a new candidate edge from robot i's original information (if none exists, $E_{cand} = \emptyset$). Moreover, the function calls a Reduce_Edge_Set sub-function that essentially prunes E_y to contain the minimum number of equality subgraph edges, such that (i) M and V_c, when calculated with respect to the pruned E_y, remain unchanged from their previous state, and (ii) the total number of edges in E_{lean} is at most $(2r - 1)$, i.e. $|E_y| + |E_{cand}| \le 2r - 1$.

The Local_Hungarian function then sets robot i's new state $\mathbb{G} = (G_{lean}, y, \gamma)$.

function Local_Hungarian $(\mathbb{G}_{tmp}, G_{orig}^i)$

(M, V_c) = maximum cardinality matching and corresponding minimum vertex cover, given (V, E_y)
if M is not a perfect matching **then**
   $e_{cand} = \text{Get\_Best\_Edge}(G_{orig}^i, y, V_c)$
   $E_{cand} = E_{cand} \cup \{e_{cand}\}$
   if $|R \setminus R_c| = |E_{cand}|$ **then** {Step 1(a) completed}
      perform Step 1(b) and Step 2
      $\gamma = \gamma + 1$
      % Prune the number of edges
      $e_{cand} = \text{Get\_Best\_Edge}(G_{orig}^i, y, V_c)$
      $E_{cand} = \{e_{cand}\}$
      $E_y = \text{Reduce\_Edge\_Set}(V, E_y, M, V_c)$
   end if
end if
$G_{lean} = (V, (E_y, E_{cand}), w_{lean})$
return $\mathbb{G} = (G_{lean}, y, \gamma)$

Now that we have stated the functions used by the *Distributed-Hungarian* algorithm, we proceed to provide a formal description of the algorithm (see Figures 9 - 8 for corresponding instances).

Algorithm *Distributed-Hungarian* (G_{orig}^i)

Initialization

choose any $j^\star \in \arg\min_{j \in P} w_{orig}^i((i, j))$
$E_y^i = \{(i, j^\star)\}$; $E_{cand}^i = \emptyset$; $w_{lean}^i = w_{orig}^i((i, j^\star))$
$G_{lean}^i = (V, (E_y^i, E_{cand}^i), w_{lean}^i)$
$y^i(i) = w_{orig}^i((i, j^\star))$; $y^i(j) = 0, \forall j \in \{\{R \setminus \{i\}\} \cup P\}$
$\gamma^i = -1$
$\mathbb{G}^i = (G_{lean}^i, y^i, \gamma^i)$

Evolution

while $\neg$ stopping criterion **do** {See Corollary 3.}
% Receive and Parse:
   $R' = \mathcal{N}_I(i, t) \cup \{i\}$, $t \in T_s$
   $S = \{\mathbb{G}^i\} \cup (\cup_{j \in R'} \{msg^j\})$
   $\mathbb{G}_{tmp} = \text{Build\_Latest\_Graph}(R', S)$
   if $\gamma \ge 0$ **then**
% Compute:
      $\mathbb{G}^i = \text{Local\_Hungarian}(\mathbb{G}_{tmp}, G_{orig}^i)$
   else
      $\mathbb{G}^i = \mathbb{G}_{tmp}$
   end if
end while

V. CONVERGENCE ANALYSIS

In this section we prove that the *Distributed-Hungarian* algorithm converges to an optimal solution in finite time. In other words, all the robots agree on a common assignment that minimizes the total cost, in finite time. We begin by proving an auxiliary lemma.

Lemma 2: For any counter value $\gamma \in \mathbb{N}_0$, there exists a (unique) vertex labeling function y_γ, and a corresponding set of equality subgraph edges, E_{y_γ}, such that for robot i's state $\mathbb{G}^i$, if $\gamma^i = \gamma$, then $y^i = y_\gamma$ and $E_y^i = E_{y_\gamma}$. Moreover, for the centralized graph $G = (V, E, w)$ (Remark 4), y_γ is a *feasible labeling* (as per Problem D - the dual of the Minimum Weight Bipartite Matching Problem).

Proof by induction (Prove for counter value 0): Recall that every robot i starts running the *Distributed-Hungarian* algorithm at time $t = t_0$, with its state $\mathbb{G}^i$ initialized as per Equations (3) - (5), to a bipartite graph G_{lean}^i, a vertex labeling function y^i and a counter value $\gamma^i = -1$. First, notice that G_{lean}^i, or more precisely E_y^i, contains exactly one, minimum weight (equality subgraph) edge from its original information G_{orig}^i. Thus, it is clear that y^i (generated using the minimum weight edge) is a feasible labeling with respect to the centralized graph $G = (V, E, w)$. Moreover, there exists a unique set of equality subgraph edges, $E_{y_0} = \cup_{i \in R} E_y^i$, that contains exactly one such edge from *every* robot, and a corresponding vertex labeling function denoted by y_0 (generated using r minimum weight edges). It is clear that y_0 is a feasible labeling with respect to the centralized graph G.

With the evolution of the algorithm, every robot i repeatedly receives the states of its incoming neighbors, and runs the Build_Latest_Graph function. If all its neighbors j have $\gamma^j = -1$, then it populates its set of equality subgraph edges E_y in its "most-updated" state $\mathbb{G}_{tmp}$, by simply merging the received equality subgraph edges, i.e., $E_y = \cup_{j \in \mathcal{N}_I(i,t) \cup \{i\}} E_y^j$.

Thus, according to the Build_Latest_Graph function, the only way the *first* robot, say i, sets its counter value $\gamma^i = 0$ is by building $\mathbb{G}_{tmp}$ with $E_y = E_{y_0}$. Moreover, another robot k sets $\gamma^k = 0$ either by building the set E_{y_0} on its own (as above), or by inheriting it from robot i (by virtue of robot i being its in-neighbor with highest counter value $\gamma^i = 0$). Iterating this argument, it follows immediately that all robots i with counter value $\gamma^i = 0$ have identical vertex labeling functions y_0, and corresponding equality subgraph edges E_{y_0}.

Assume true for counter value n: Assume that all robots with counter value n have identical vertex labeling functions, and identical equality subgraph edges, denoted by y_n and E_{y_n} respectively. Moreover, assume that y_n is a *feasible labeling* with respect to the centralized graph G.

Prove for counter value $n + 1$: Given y_n, E_{y_n}, and consequently a maximum cardinality matching M_n and corresponding minimum vertex cover V_{c_n}, there exists a unique set E_{cand_n} that comprises of *exactly one edge* from each uncovered robot in $R \backslash R_{c_n}$ (going to an uncovered target in $P \backslash P_{c_n}$). Such an edge is uniquely determined by the Get_Best_Edge function, identical to the process of selecting candidate edges in the centralized Hungarian_Method. Thus, E_{cand_n} corresponds to a *completed* Step 1(a) of the Hungarian_Method. As such, if any robot i constructs a "most-updated" state $\mathbb{G}_{tmp}$ with the unique set E_{cand_n}, then by construction, its Local_Hungarian function proceeds to perform Step 1(a), Step 1(b) and Step 2 of the Hungarian_Method on $\mathbb{G}_{tmp}$, resulting in a counter value of $n + 1$, and an updated vertex labeling y_{n+1} (with corresponding set of equality subgraph edges $E_{y_{n+1}}$). Most importantly, the two-step iteration of the Local_Hungarian function satisfies the conditions in Lemma 1, proving that the updated y_{n+1} is *still* a feasible labeling with respect to the centralized graph G.

As the algorithm evolves, every robot i with counter value n includes in its state, a candidate edge $e_{cand} \in E_{cand_n}$ from its original information G_{orig}^i, using the Get_Best_Edge function. Upon receiving the states of incoming neighbors, if all its neighbors j have $\gamma^j \leq n$, then robot i populates its set of candidate edges in $\mathbb{G}_{tmp}$, by simply merging the received candidate edges from *only* those robots, with highest counter value, i.e., n.

Thus, according to the Build_Latest_Graph and the Local_Hungarian function, the only way the *first* robot, say i, sets $\gamma^i = n + 1$ is by building $\mathbb{G}_{tmp}$ with $E_{cand} = E_{cand_n}$. Similar to the argument for counter value 0, another robot k sets $\gamma^k = n + 1$ either by building the set E_{cand_n} on its own (as above), or inheriting the latest information directly from robot i (by virtue of robot i being its in-neighbor with highest counter value $\gamma^i = n + 1$), thereby concluding the proof. ∎

Corollary 1: If two robots i and j have identical counter values, then with respect to the graphs $(V, E_y^i) = (V, E_y^j)$, they have identical maximal matchings, i.e.,

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

Proof: Let $\gamma^i = \gamma^j = n$, for some $n \in \mathbb{N}_0$. Since the counter value n corresponds to a unique set of equality subgraph edges E_{y_n}, the maximal matching found from within the set of such edges is also unique. Thus, robots i and j have identical maximal matchings, denoted by M_n. ∎

Corollary 2: If robot i's counter value is higher than robot j's counter value, i.e. $\gamma^i > \gamma^j$, then with respect to the graphs (V, E_y^i) and (V, E_y^j), one of the following is true,

- robot i's maximal matching, M^i, is greater in size than robot j's maximal matching, M^j, i.e. $|M^i| > |M^j|$;
- robots i and j have the same maximal matching, i.e. $M^i = M^j$, but in the context of their corresponding minimum vertex covers V_c^i and V_c^j, robot i has more covered vertices in R than robot j, i.e., $|R_c^i| > |R_c^j|$.

Proof: Let $\gamma^i = p$, and $\gamma^j = q$, where $p, q \in \mathbb{N}_0$, $q < p$. From the proof of Lemma 2, we know that for a counter value to increment to say $n + 1$, there must have existed at least one robot with counter value n, at some previous iteration of the *Distributed-Hungarian* algorithm. In other words, for every $n \in \{q, ..., p\}$, there *existed* a robot with counter value n, a corresponding feasible vertex labeling function y_n, a set of equality subgraph edges E_{y_n}, and consequently, a maximal matching M_n, during some previous iteration of the algorithm. By construction, since every counter value update that occurs from n to $n + 1$, $n \in \{q, ..., p\}$ corresponds to a two-step iteration of the Hungarian_Method, then (statement (ii) of) Lemma 1 holds. Iteratively, the proof follows. ∎

Theorem 2: Given a set of robots R, a set of targets P, and a time-varying communication graph $\mathcal{G}_c(t)$, $t \in \mathbb{R}_{\geq 0}$, satisfying Assumption 1, assume every robot $i \in R$ knows (R, P, c^i). If the robots execute the *Distributed-Assignment* algorithm, there exists a finite time T_f such that all robots converge to a common assignment $\hat{M}$, which is an optimal solution of the (centralized) assignment problem, i.e., $\hat{M} \in \mathcal{M}$.

Proof: We proceed to prove the theorem in the following three steps: (i) for every robot i in R, the counter value γ^i evolves as a monotone, non-decreasing sequence that converges in finite time, (ii) once all the counter values have converged, they must be at the same value, and (iii) at steady-state, a common, perfect matching corresponding to the optimal assignment is computed.

By the connectivity assumption (Assumption 1), there exists a finite time interval in which at least one robot i is able to construct a "most-updated" state $\mathbb{G}_{tmp}$ that contains *enough candidate edges* to perform Step 1 and Step 2 in the Local_Hungarian function, thereby incrementing its counter value from n to $n + 1$. Moreover, from Lemma 2, we know that such a counter value increment satisfies the conditions in Lemma 1. As such, we can use the proof-sketch in Remark 3 to show that there exists only a finite number of such counter value increments (worst case $\mathcal{O}(r^2)$, with $r = |R|$) before
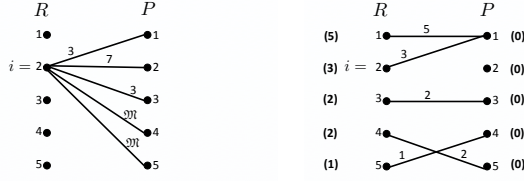
(a) Robot 2's original information as the weighted, bipartite graph $G^2_{orig} = (V, E^2_{orig}, w^2_{orig})$.

(b) Robot 2's temporary, most-updated state $\mathbb{G}_{tmp} = (G_{lean}, y, \gamma)$ with $\gamma = 0$. E_{lean} contains the equality subgraph edges E_y (black edges) and the candidate edges E_{cand} (green edges, none in this case).
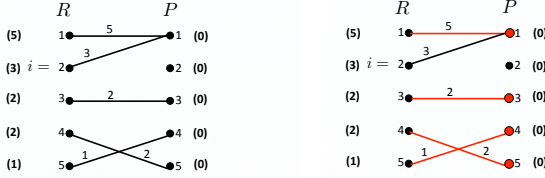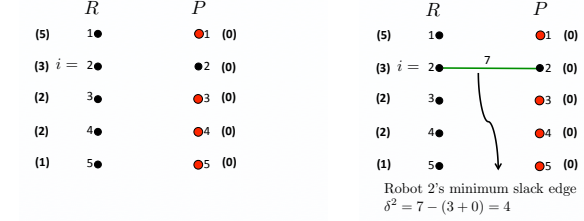
(c) The isolated set of equality subgraph edges E_y in $\mathbb{G}_{tmp}$.

(d) Given E_y, a maximum cardinality matching M_y (red edges), and a corresponding minimum vertex cover $V_c = (R_c, P_c)$ (red vertices).

Fig. 4: An instance of the *Distributed-Hungarian* algorithm. (a) Robot 2's original information, (b) Example output for the Build_Latest_Graph function, (c) & (d) Within the Local_Hungarian function: isolated set E_y, and corresponding M_y and V_c respectively.

(a) The isolated set of candidate edges E_{cand} in $\mathbb{G}_{tmp}$ (none in this case).

(b) Given V_c, Robot 2's set of candidate edges (green edges, only one in this case) using its original information G^2_{orig}. Exactly one edge with minimum slack δ is chosen for inclusion in E_{cand}.

(c) The updated set of candidate edges E_{cand} (green edges), combining the edges from Figures 5b and 5a. Note that the number of edges is *not sufficient* to proceed.

(d) Robot 2's outgoing message $msg^2 = \mathbb{G}^2$, comprising of the graph $G^2_{lean} = (V, E^2_{lean}, w^2_{lean})$, the vertex labeling function y^2, and the counter value $\gamma^2 = 0$.

Fig. 5: Contd. from Figure 9. (a), (b) & (c) Within the Local_Hungarian function: isolated set E_{cand}, output of the Get_Best_Edge function, updated set E_{cand} with *insufficient* edges, respectively and (d) Output of the Local_Hungarian function, i.e. Robot 2's outgoing message $msg^2 = \mathbb{G}^2$.
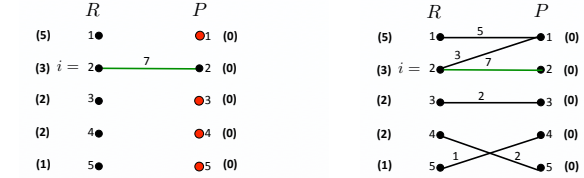
which the matching found is perfect. Moreover, by construction, once a robot computes a perfect (and hence optimal) matching, the counter value stops increasing. Therefore, each so-called counter-sequence must converge in finite time.

Once all the counter-sequences have converged, they must be at the same value. Indeed, if this is not the case, by Assumption 1, there must exist two robots, say i and j at some time $t \in T_s$, such that $\gamma^i < \gamma^j$, and j is an in-neighbor to i. By construction, robot i would receive a message from robot j and set its counter value $\gamma^i = \gamma^j$, contradicting the fact that all the sequences have converged.

Next, at some time t, let $\gamma^i = \gamma^j = n$, $\forall i, j \in R$. Then by Lemma 2 and Corollary 1, we know that all robots have the same vertex labeling function y_n that is also a *feasible labeling* for the centralized graph G, and the same maximal matching M_n respectively. Thus, if M_n is a perfect matching, then by the Kuhn-Munkres Theorem (Theorem 1), it is an optimal solution to the centralized assignment problem. Suppose, by contradiction, M_n is not a perfect matching. In that case, every robot contributes a candidate edge to its state (if such an edge exists), and sends it to its outgoing neighbors. As mentioned previously, there exists a finite time interval in which at least one robot i is able to construct a "most-updated" state $\mathbb{G}_{tmp}$ that contains *enough candidate edges* to update its counter value. This contradicts the fact that the counter-sequences of

all the robots have converged and concludes the proof. ∎

Note that since we describe a synchronous implementation of the *Distributed-Hungarian* algorithm, we can take the results from Theorem 2 one step further and quantify both, the stopping criterion and convergence time, in terms of *iterations* (communication rounds) of the algorithm. As such, we provide the following auxiliary lemma,

Lemma 3 (Information Dispersion): If robot i sends its information to its outgoing neighbors (which in turn propagate the information forward, and so on), then within a maximum of $(r - 1)$ iterations (communication rounds) of the *Distributed-Hungarian* algorithm, robot i's information reaches every other robot in the network.

Proof sketch: By Assumption 1, at every iteration (time instant $t \in T_s$) of the *Distributed-Hungarian* algorithm, the underlying dynamic, directed communication network $\mathcal{G}_c(t)$ is strongly connected. Such a criterion implies that at every time $t \in T_s$, there exists a *directed path* (sequence of edges) from every robot to every other robot in the network. Thus, on the first iteration of the *Distributed-Hungarian* algorithm, at least one robot j is robot i's out-neighbor and receives its information. It follows that on each consequent iteration, at least one *new* robot is added to the set of robots that have already received robot i's information, by virtue of being an
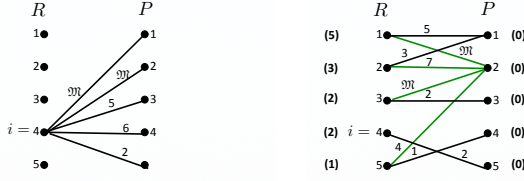
(a) Robot 4's original information as the weighted, bipartite graph $G^4_{orig} = (V, E^4_{orig}, w^4_{orig})$.

(b) Robot 4's temporary, most-updated state $\mathbb{G}_{tmp} = (G_{lean}, y, \gamma)$, with $\gamma = 0$. E_{lean} contains the equality subgraph edges E_y (black edges) and the candidate edges E_{cand} (green edges).
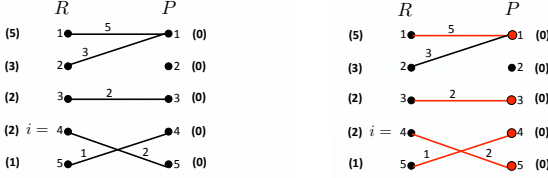
(c) The isolated set of equality subgraph edges E_y in $\mathbb{G}_{tmp}$.

(d) Given E_y, a maximum cardinality matching M_y (red edges), and a corresponding minimum vertex cover $V_c = (R_c, P_c)$ (red vertices).

Fig. 6: Another instance of the *Distributed-Hungarian* algorithm. (a) Robot 4's original information, (b) Example output for the Build_Latest_Graph function, (c) & (d) Within the Local_Hungarian function: isolated set E_y, and corresponding M_y and V_c respectively.

(a) The isolated set of candidate edges E_{cand} in $\mathbb{G}_{tmp}$.

(b) Given V_{c_y}, Robot 4's set of candidate edges (green edges) using its original information G^4_{orig}. Exactly one edge (with corresponding minimum slack δ) is chosen for inclusion in E_{cand}.

(c) Step 1(a): The updated set of candidate edges E_{cand} (green edges), combining the edges from Figures 7b and 7a. Step 1(b): Since the number of edges is sufficient to proceed, the minimum slack edge is identified.

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

Fig. 7: Contd. from Figure 6. Within the Local_Hungarian function: (a) Isolated set E_{cand}, (b) Output of the Get_Best_Edge function, (c) Updated set E_{cand} with *sufficient* edges (Step 1(a)), and identified minimum slack edge (Step 1(b)) and (d) Updated y (Step 1(b) contd.).
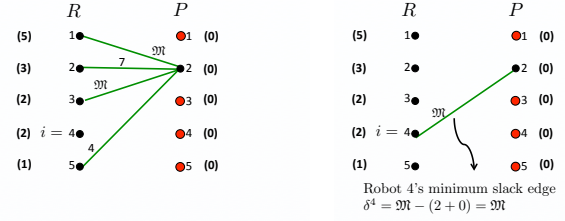
out-neighbor to at least one of them, thus concluding the proof-sketch. ∎

Corollary 3 (Stopping Criterion): Suppose robot i finds a perfect matching on some iteration of the *Distributed-Hungarian* algorithm. Then robot i can stop sending its corresponding message msg^i after $(r - 1)$ iterations (communication rounds).
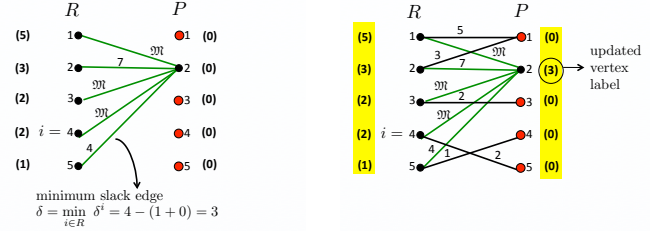
Proof: Let M^i denote the perfect matching found by robot i. As discussed in the proof of Theorem 2, M^i is also optimal with respect to the centralized assignment problem, and can be denoted by $\hat{M}$. Thus, using Lemma 3, within a maximum of $(r-1)$ iterations, every robot in the network will receive 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 (Convergence Time): A common, optimal solution $\hat{M} \in \mathcal{M}$ is found in $\mathcal{O}(r^3)$ iterations (communication rounds) of the *Distributed-Hungarian* algorithm.

Proof: From Theorem 2, we know that the convergence of a counter-sequence implies that a common, optimal solution has been found (where the number of counter value increments cannot exceed $\mathcal{O}(r^2)$). Moreover, from Lemma 3, we know that within $(r - 1)$ iterations of the algorithm, the highest counter value among all robots is incremented (irrespective of

the robot it belongs to). Thus, a common, optimal solution $\hat{M}$ is found in $\mathcal{O}(r^3)$ iterations. ∎

Remark 6 (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 $\mathfrak{M}$. □

Remark 7 (Message Size): Recall that a robot i's message, msg^i, comprises of a sparse graph $G^i_{lean} = (V, E^i_{lean}, w^i_{lean})$ with at most $(2r - 1)$ edges, a vertex labeling function $y^i : V \to \mathbb{R}$, and a counter value, $\gamma^i \in \mathbb{Z}$. Edges in E_{lean} can be encoded with $\lceil \frac{1}{4} \log_2(r) \rceil$ bytes each, while edge weights and vertex labels can be encoded with 2 bytes each (approximating a real number as a 16-bit floating point value). Moreover, since the counter value represents the number of two-step iterations (maximum r^2 as per Remark 3), it can be encoded as an integer with $\lceil \frac{1}{4} \log_2(r) \rceil$ bytes. Thus, at each iteration (communication round) of the algorithm, $((2r) \cdot (4 + \lceil \frac{1}{4} \log_2(r) \rceil) - 2)$ bytes are sent out by each robot. □

Remark 8: [Asynchronous Implementation] As mentioned previously, the connectivity assumption (Assumption 1) can be relaxed by requiring the communication graph to be only *jointly* strongly connected over some time period T_c, lending towards an asynchronous framework. Also, in our
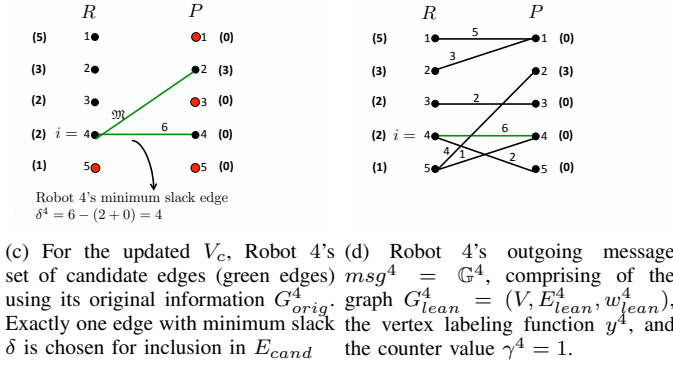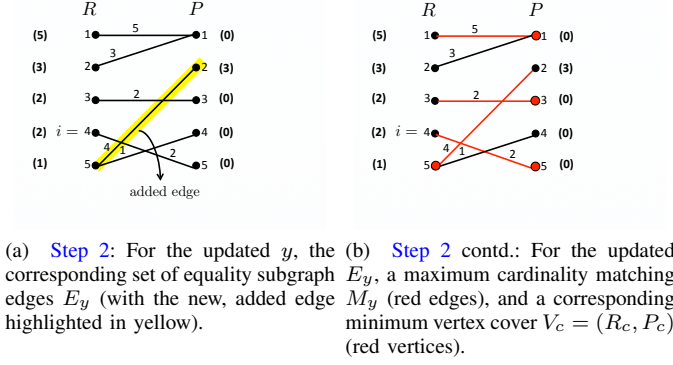
(a) **Step 2**: For the updated y, the corresponding set of equality subgraph edges E_y (with the new, added edge highlighted in yellow).

(b) **Step 2** contd.: For the updated E_y, a maximum cardinality matching M_y (red edges), and a corresponding minimum vertex cover $V_c = (R_c, P_c)$ (red vertices).

(c) For the updated V_c, Robot 4's set of candidate edges (green edges) using its original information G_{orig}^4. Exactly one edge with minimum slack δ is chosen for inclusion in E_{cand}

(d) Robot 4's outgoing message $msg^4 = \mathbb{G}^4$, comprising of the graph $G_{lean}^4 = (V, E_{lean}^4, w_{lean}^4)$, the vertex labeling function y^4, and the counter value $\gamma^4 = 1$.

Fig. 8: Contd. from Figure 7. (a), (b) & (c) Within the Local_Hungarian function: updated set E_y (Step 2), corresponding M_y and V_c (Step 2 contd.), and output of the Get_Best_Edge function respectively, and (d) Output of the Local_Hungarian function, i.e. Robot 4's outgoing message $msg^4 = \mathbb{G}^4$.
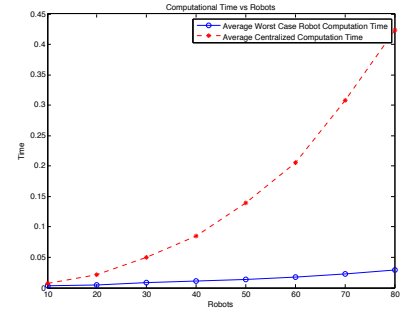
implementation of the *Distributed-Hungarian* algorithm, time t is universal time, and as such, need not be explicitly known by the robots. Due to this independence, the *Distributed-Hungarian* algorithm can indeed run asynchronously under the joint connectivity assumption. The proof sketch for such an implementation is similar to the synchronous case explained earlier in this section, and relies on the fact that within a maximum time interval T_c, there exists a time dependent directed path between every pair of robots, thus ensuring subsequent convergence. □

A. Simulation Experiments

To assess the performance of our proposed *Distributed-Hungarian* algorithm in practice, we performed simulation experiments on multiple instances of the LSAP with varying problem sizes, and plotted the average number of iterations required for convergence. In particular, the simulation experiments were performed in MATLAB, and executed on a PC with an Intel Quad Core i5, 3.3GHz CPU and 16GB RAM. For every r (total number of robots) varying from 5 to 160, we performed 20 runs of the *Distributed-Hungarian* algorithm, over randomly generated $(r \times r)$ cost matrices with cost $c_{i,j} \in (0, 1000)$, and a strongly connected, communication network with *dynamic* incoming and outgoing edges between
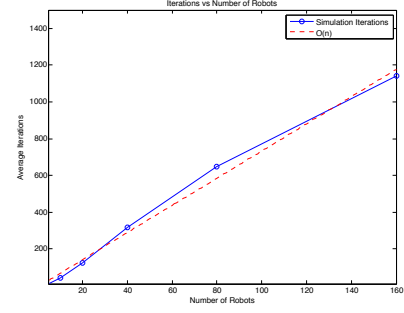
(a) Average number of iterations required for convergence, versus the number of robots.

(b) Average, worst-case computational time of an iteration, versus the number of robots.

Fig. 9: Simulation experiments characterizing different aspects of the *Distributed-Hungarian* algorithm.

robots at each time instant of the synchronous implementation.

As seen in Figure 9a, the average number of iterations required for convergence is well under $\mathcal{O}(r^3)$ (worst-case bound). Moreover, to qualify the computational load on a robot at any given time, on each run of the *Distributed-Hungarian* algorithm, we observed the iteration that took the maximum computational time (across all robots), and plotted the average of all 20 runs, against the computational time of the corresponding centralized Hungarian_Method (Figure 9b). The experiments support the applicability of the *Distributed-Hungarian* algorithm to varying sized teams of mobile robots in practice. As such, we proceed to describe the motivating application in this paper, which provides an intuitive and immediate testbed for demonstrating our proposed algorithm.

VI. A MOTIVATING APPLICATION: DYNAMIC SPATIO-TEMPORAL MULTI-ROBOT ROUTING

To demonstrate the distributed algorithm central to this paper, we consider multi-robot routing as a motivating application. In particular, we consider a special class of routing problems called "spatio-temporal routing problems", previously introduced in [37]. In such routing problems, each target is associated with a *specific time instant* at which it requires servicing. Additionally, each target, as well as each robot, is associated with one or more skills, and a target is serviceable by a robot *only if* the robot has a skill in common with the skill set of that target (or in other words, the robot is authorized
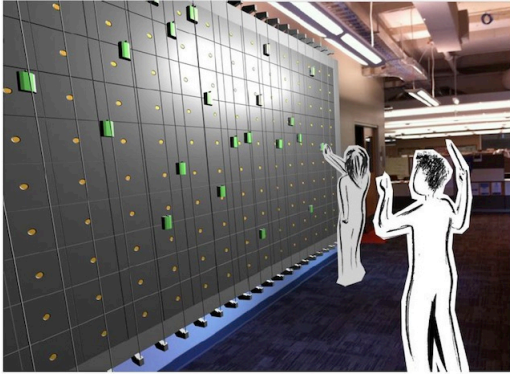
Fig. 10: A rendering of the *Robot Orchestral Floor* concept (viewed in the picture, as a vertical wall instead of a floor).

to service the target). As shown in [37], task assignment is a convenient framework to attack such spatio-temporal routing problems.

In this section, we briefly discuss the extension of our previous work in [37] on spatio-temporal routing from a static, centralized solution to its dynamic and distributed counterpart. More specifically, for a series of spatio-temporal requests, the robots cooperatively determine their routes online by using the *Distributed-Hungarian* algorithm.

The online scheme we propose is based on a simple, widely applied idea, where assignments are solved iteratively between consecutive time instants. This scheme provides an effective framework for incorporating our distributed algorithm towards dynamic spatio-temporal routing. We further illustrate this setup in a musical environment through a novel, "multi-robot orchestral" framework. We acknowledge that we are not trying to find the most "optimal " or "efficient" solution to the routing problem itself. Instead, we are interested in demonstrating the applicability of our proposed algorithm, in a practical, and intuitive setting.

Each robot can play one or more instruments (essentially, a piano, a guitar and drums), and a piece of music can be interpreted as a series of spatio-temporal requests on the so-called *Robot Orchestral Floor* - a music surface where planar positions correspond to distinct notes of different instruments (see Figure 10 for an illustration).

A user (acting similar to a "conductor") can change the piece of music in real-time, while the robots adapt their routes accordingly, to incorporate the changes. The user interacts with the team of robots by means of a tablet interface through which it broadcasts the spatio-temporal requests.[8]

A. Overview of the Methodology

For convenience, we assume that the minimum time difference between two timed positions is always greater than the time needed by the robots to solve an instance of the

[8]This paper considers the generalized version of the routing problem that associates a *set of skills or instruments* with each spatio-temporal request. However, for the purpose of musical demonstration on the orchestral floor, a *single* skill (instrument) is associated with each request (w.l.o.g.).

Distributed Hungarian algorithm, and to reach their assigned positions.

In the following paragraph, we put forth the two key ideas, central to the scheme described above.

- *Distributed Aspect:* Given a Score, the robots determine routes by iteratively solving assignments using the *Distributed-Hungarian* algorithm, between successive time instants. Each instance of an assignment can be formulated as an unbalanced Linear Sum Assignment Problem (0-1 linear program) [8], using a mapping $l(p, \alpha)$ as follows,

For the consecutive time instants t_i to t_{i+1}, given the quintuple $(Sc_i, R, M_{pos}, M_{rbt})$, and the function $P_{rbt} : R \rightarrow \mathbb{R}^2$, denoting the planar positions of the robots[9], find l such that:

$$\min_l \sum_{p \in R} \sum_{\alpha \in 1}^{|Sc_i|} ||P_{i,\alpha} - P_{rbt}(p)|| \, l(p, \alpha) \quad (6)$$

subject to:

$$l(p, \alpha) \in \{0, 1\} \quad (7)$$

$$\sum_{p \in R} l(p, \alpha) = 1, \quad \forall \, \alpha \in \{1, ..., |Sc_i|\} \quad (8)$$

$$\sum_{\alpha \in \mathcal{A}_i} l(p, \alpha) \leq 1, \quad \forall \, p \in R \quad (9)$$

$$l(p, \alpha) = 1 \Rightarrow M_{rbt}(p) \cap M_{pos}((P_{i,\alpha}, t_i)) \neq \emptyset \quad (10)$$

where $l(p, \alpha)$ represents the individual assignment of robot $p \in R$ to timed position $(P_{i,\alpha}, t_i) \in Sc_i$, and is 1 if the assignment is done, and 0 otherwise.

Note that for employing the *Distributed-Hungarian* algorithm, we view the above stated LSAP in graph theoretic terms, as the equivalent *Minimum Weight Bipartite Matching Problem (P)* from II-A (we assume that the underlying time-varying communication graph, induced as the robots execute their paths, satisfies Assumption 1). Moreover, the robots solve assignments between *future* consecutive time instants, while *simultaneously* executing routes that they have already determined (see Figure 11a).

- *Dynamic Aspect:* A user can dynamically modify the Score as follows:
  i Add a timed position with a corresponding skill set (add a note of an instrument (piano or guitar), or a beat of a drum, to be played at a particular time instant).
  ii Remove a timed position (remove a note of an instrument (piano or guitar), or a beat of a drum, from a particular time instant).
  iii Modify the skill set of a timed position (substitute the instrument of a note (from a piano to a guitar, or vice versa), or replace one kind of drum with another, at a particular time instant).

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

[9]Under the iterative scheme, each robot's planar position is a *previously assigned* timed position at some time instant $t_j \leq t_i$ (unless $i = 0$).
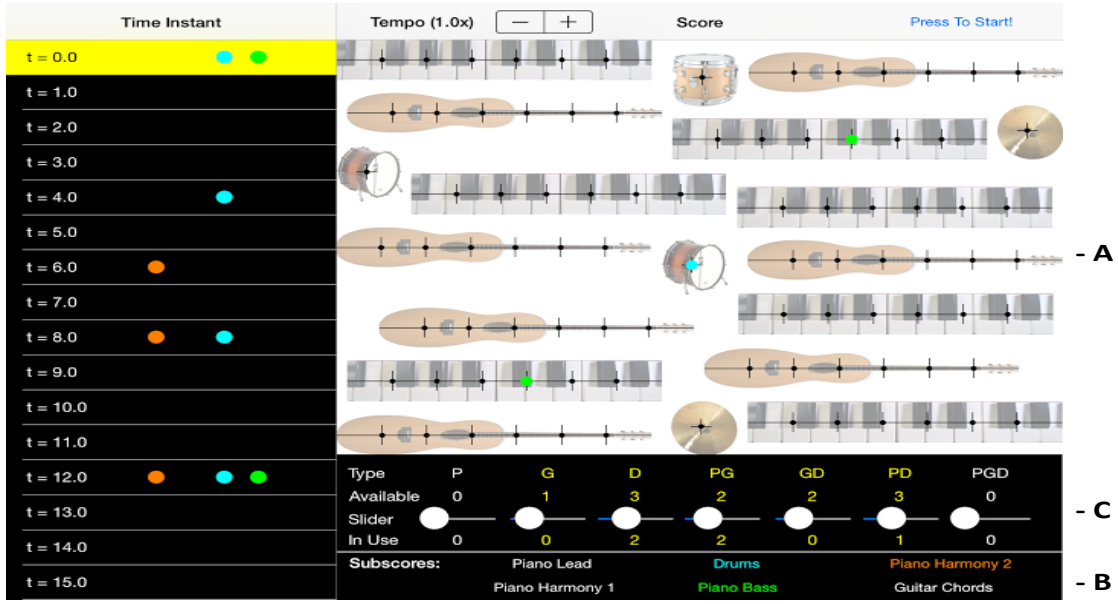
Fig. 12: The tablet interface for modifying and broadcasting changes to the Score. **A**: A simulated example of the *Robot Orchestral Floor* comprising of unique positions corresponding to piano and guitar notes, as well as drum beats. **B**: Pre-defined sub-scores for an instrumental version of "The Final Countdown". **C**: A feature for either selecting, or setting the available number of robots (R, M_{rbt}).

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 11b - 11d, we provide examples of three different cases that can occur, when a particular dynamic modification is received.

As mentioned previously, a user issues dynamic modifications through a user-interface. We assume that such an interface has knowledge of the Score and the robots, i.e. $(Sc_i, R, M_{pos}, M_{rbt})$, and is able to broadcast the modifications to the robots in real time. Since we assume that the timed positions in the Score are sufficiently apart in time, the interface does not allow a user to modify the Score unless the modification occurs after a pre-specified (conservative) time duration (depicted by the red regions in Figures 11b - 11d). Moreover, the interface does not allow modifications that violate feasibility (in that, the available number of robots do not fall short), as per our results in [37][10].

B. The Multi-Robot Orchestra

In this section, we apply the theory developed so far, towards enabling multiple robots to execute different musical pieces (presented to them as Scores). To this end, we simulated a version of the *Robot Orchestral Floor* in MATLAB, instrumented to include piano, guitar and drum sounds (see A in Figure 12). In addition to the simulated floor, we developed

a graphical user-interface (GUI) (Figure 12) that allows a user to create, and administer changes to a Score on the simulated floor. The user-interface is developed on a tablet, 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 12). 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 spatio-temporal routing, either the user selects, or is given the number of robots available for use (see C in Figure 12). Moreover, the user creates an initial Score using the iPad interface. We assume that all robots are initially positioned along a vertical edge of the floor. Once the user hits the start button, the iPad broadcasts this Score to the team of robots. From this point onwards, the routes of the robots are determined and executed in real time, while the iPad broadcasts changes to the Score, as and when a user decides to modify it.

We implemented the multi-robot orchestra in both simulation and hardware environments (Figures 13). The hardware

---

[10]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.
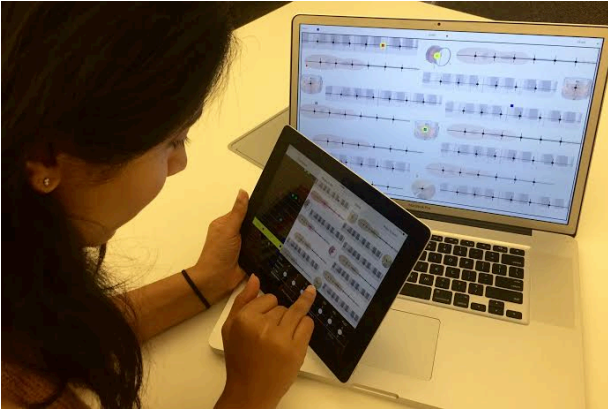
(a) A user interacting with simulated robots through the orchestral floor tablet interface.
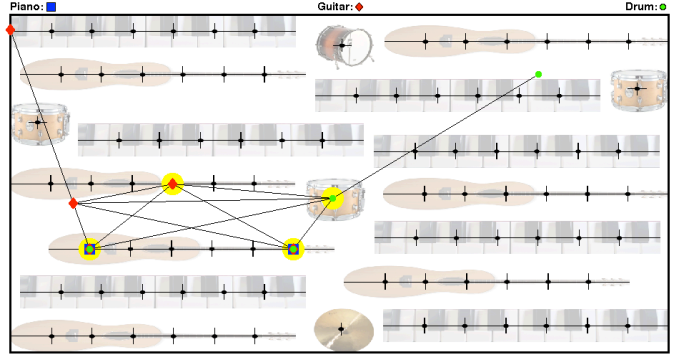
(b) An instance of simulated robots (geometric shapes - diamonds, circles and squares), performing the Score by executing dynamic spatio-temporal routing (the black lines denote the underlying dynamic communication network, required for a distributed implementation).
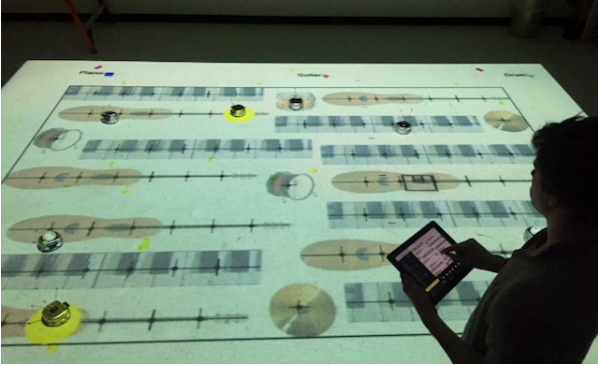
(c) A user interacting with actual robots through the orchestral floor tablet interface.
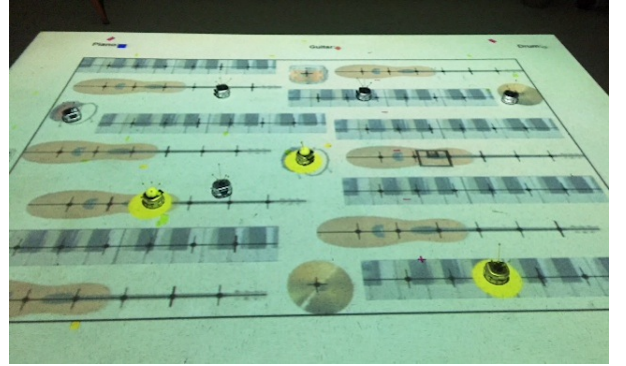
(d) An instance of actual robots, performing the Score by executing dynamic spatio-temporal routing.

Fig. 13: Simulation (https://www.youtube.com/watch?v=L-PSPy9O_BE), and hardware (https://www.youtube.com/watch?v=z7SiivWvZLc) implementation of the multi-robot orchestra, performing "The Final Countdown" on the *Robot Orchestral Floor*, with a user "conducting" (modifying) the Score through the tablet interface. In both cases, when a particular robot reaches a timed position on the orchestral floor, its is highlighted by a light (yellow) circle, and the corresponding sound of the note/beat is generated.

implementation was conducted in the Georgia Robotics and Intelligent Systems Laboratory, where the indoor facility is equipped with a motion capture system which yields real time accurate data for all tracked objects, and an overhead projector is used for embedding algorithm/environment-specific information (see [34] for a similar hardware setup). We used Khepera III miniature robots by K Team as our hardware ground robots. In both cases (simulation and hardware), the instant a robot reached a timed position (a note of an instrument, or a beat of a drum, specified at a particular time instant) on the orchestral floor, it was encircled by a light (yellow) circle and the corresponding sound of the note (or beat in the case of drums) was generated. In this manner, we enabled multiple robots to perform a real-time rendition of "The Final Countdown".

## VII. Conclusion and Future Directions

This paper provides a distributed version of the *Hungarian Method* for solving the well known Linear Sum Assignment Problem (LSAP). The proposed algorithm allows a team of robots to cooperatively compute the optimal solution to the LSAP, without any coordinator or shared memory. We prove that under a synchronous implementation, all robots converge to a *common* optimal assignment within $\mathcal{O}(r^3)$ iterations. By running simulation experiments over multiple instances of the LSAP with varying problem sizes, we show that the average number of iterations for convergence is much smaller than the theoretic worst-case bound of $\mathcal{O}(r^3)$. Moreover, we show that the computational load per robot is minor in comparison to the centralized Hungarian_Method, since the robots perform only sub-steps of the centralized algorithm, at each iteration of the proposed algorithm.

To demonstrate the theory developed in this paper, we extend our proposed algorithm to solving a class of "spatio-temporal" multi-robot routing problems, considered under a distributed and dynamic setting. In essence, the robots find online, sub-optimal routes by solving a sequence of assignment problems iteratively, using the proposed distributed algorithm
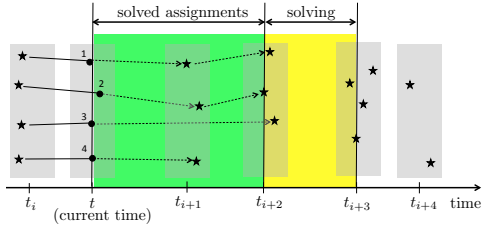
(a) *Future* portions of the routes of the robots (between time instants t_{i+2} to t_{i+3}) are currently being determined, while the robots execute the portion of the routes that have already been determined (up until t_{i+2}).
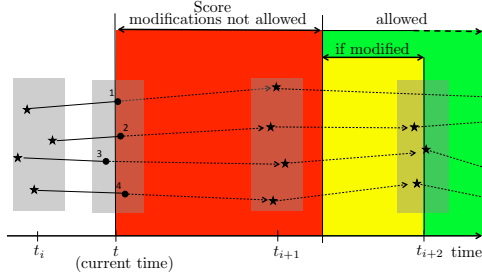
(b) Case 1: If at the current time t, a modification is received, specified at a time instant in the depicted time interval (yellow), *all* robots retain their routes up until t_{i+1} and begin recalculating their routes, from their positions at t_{i+1} onwards.
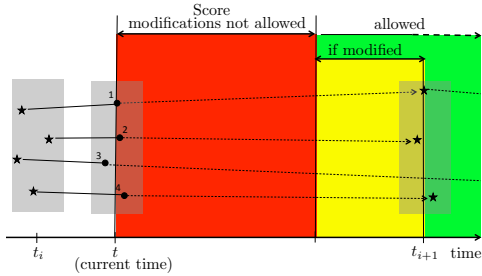
(c) Case 2: If at the current time t, a modification is received, specified at a time instant in the depicted time interval (yellow), *all* robots begin recalculating their routes, from their current positions at time t onwards.
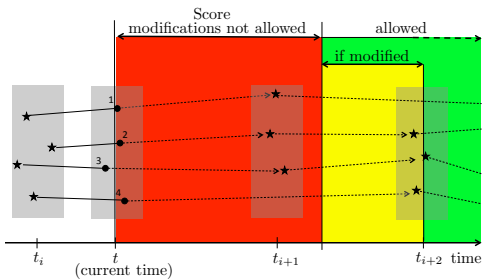
(d) Case 3: If at the current time t, a modification is received, specified at a time instant in the depicted time interval (yellow), *only* robots 1, 2 and 3 retain their routes up until t_{i+1}. The robots begins recalculating their routes from their positions at t_{i+1} onwards, however robot 4 begins recalculating its route from its current position at time t onwards.

Fig. 11: Robots (circles) at time t, and timed positions (stars) grouped in grey boxes representing specific time instants.

for each instance. As a motivating application and concrete experimental test-bed, we develop the "multi-robot orchestral" framework, where spatio-temporal routing is musically interpreted as "playing a series of notes at particular time instants" on a so-called orchestral floor (a music surface where planar positions correspond to distinct notes of different instruments). Moreover, we allow a user to act akin to a "conductor", modifying the music that the robots are playing in real time through a tablet interface. Under such a framework, we perform simulations and hardware experiments that showcase our algorithm in a practical setting.

An interesting future direction that we are currently exploring, is the interpretation of the Hungarian Method as a primal algorithm (as opposed to its native, dual form), and its subsequent redesign in a distributed setting. Essentially, the primal algorithm would provide a constantly improving *feasible* assignment at every iteration. We hypothesize that such an algorithm, when redesigned in a distributed setting, could be more robust and simple than the current approach described in this paper, though slower to converge. Distributed implementations of both primal and dual versions of the Hungarian Method would yield interesting technical comparisons in the assignment literature.

REFERENCES

[1] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," Econometrica: journal of the Econometric Society, pp. 53–76, 1957.
[2] C. Papadimitriou and K. Steiglitz, Combinatorial optimization: algorithms and complexity. Courier Dover Publications, 1998.
[3] D. Pentico, "Assignment problems: A golden anniversary survey," European Journal of Operational Research, vol. 176, no. 2, pp. 774 – 793, 2007.
[4] R. Burkard, M. Dell'Amico, and S. Martello, Assignment problems. Society for Industrial and Applied Mathematics, 2009.
[5] D. Bertsekas, "A new algorithm for the assignment problem," Mathematical Programming, vol. 21, no. 1, pp. 152–171, 1981.
[6] 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.
[7] M. Balinski, "Signature methods for the assignment problem," Operations research, vol. 33, no. 3, pp. 527–536, 1985.
[8] R. E. Burkard and E. Cela, Linear assignment problems and extensions. Springer, 1999.
[9] H. Kuhn, "The hungarian method for the assignment problem," Naval Research Logistics, vol. 2, no. 1-2, pp. 83–97, 1955.
[10] M. J. Matarić, G. S. Sukhatme, and E. H. Østergaard, "Multi-robot task allocation in uncertain environments," Autonomous Robots, vol. 14, no. 2-3, pp. 255–263, 2003.
[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. Ji, S.-i. Azuma, and M. Egerstedt, "Role-assignment in multi-agent coordination," International Journal of Assistive Robotics and Mechatronics, 2006.
[13] S. Smith and F. Bullo, "Target assignment for robotic networks: Asymptotic performance under limited communication," in Proceedings of the American Control Conference, 2007, pp. 1155–1160.
[14] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," Proceedings of the IEEE, vol. 94, no. 7, pp. 1257–1270, 2006.

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

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

[17] M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing," in Robotics: Science and Systems, 2005, pp. 343–350.

[18] M. Nanjanath and M. Gini, "Repeated auctions for robust task execution by a robot team," Robotics and Autonomous Systems, vol. 58, no. 7, pp. 900–909, 2010.

[19] M. Zavlanos, L. Spesivtsev, and G. Pappas, "A distributed auction algorithm for the assignment problem," in IEEE Conference on Decision and Control, 2008, pp. 1212–1217.

[20] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in IEEE International Conference on Robotics and Automation, 2008, pp. 128–133.

[21] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," IEEE Transactions on Robotics, vol. 25, no. 4, pp. 912–926, 2009.

[22] L. Luo, N. Chakraborty, and K. Sycara, "Multi-robot assignment algorithm for tasks with set precedence constraints," in IEEE International Conference on Robotics and Automation, 2011, pp. 2526–2533.

[23] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," The International Journal of Robotics Research, vol. 25, no. 3, pp. 225–241, 2006.

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

[25] A. Tahbaz-Salehi and A. Jadbabaie, "On consensus over random networks," in 44th Annual Allerton Conference. Citeseer, 2006.

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

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

[28] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," Automatica, vol. 48, no. 9, pp. 2298–2304, 2012.

[29] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in Trends in Applied Intelligent Systems. Springer, 2010, pp. 721–730.

[30] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous vehicle-target assignment: A game-theoretical formulation," Journal of Dynamic Systems, Measurement, and Control, vol. 129, no. 5, pp. 584–596, 2007.

[31] J. R. Marden and A. Wierman, "Distributed welfare games," Operations Research, vol. 61, no. 1, pp. 155–168, 2013.

[32] L. Johnson, H.-L. Choi, S. Ponda, and J. P. How, "Allowing non-submodular score functions in distributed task allocation," in IEEE Conference on Decision and Control. IEEE, 2012, pp. 4702–4708.

[33] L. B. Johnson, H.-L. Choi, and J. P. How, "Hybrid information and plan consensus in distributed task allocation," in AIAA Guidance, Navigation, and Control (GNC) Conference, 2013, p. 4888.

[34] S. Omidshafiei, A.-A. Agha-Mohammadi, Y. F. Chen, N. K. Üre, J. P. How, J. L. Vian, and R. Surati, "Mar-cps: Measurable augmented reality for prototyping cyber-physical systems," in AIAA Infotech@ Aerospace, 2015, p. 0643.

[35] S. Giordani, M. Lujak, and F. Martinelli, "A distributed multi-agent production planning and scheduling framework for mobile robots," Computers & Industrial Engineering, vol. 64, no. 1, pp. 19–30, 2013.

[36] S. Chopra and M. Egerstedt, "Spatio-temporal multi-robot routing," Automatica, 2015.

[37] ——, "Heterogeneous multi-robot routing," Proceedings of the American Control Conference, 2014.

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

[39] J. Hopcroft and R. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," SIAM Journal on computing, vol. 2, no. 4, pp. 225–231, 1973.

[40] A. Frank, "On kuhn's hungarian method—a tribute from hungary," Naval Research Logistics (NRL), vol. 52, no. 1, pp. 2–5, 2005.

Smriti Chopra Smriti Chopra is a graduate of the Georgia Institute of Technology. She received a Ph.D degree in Electrical and Computer Engineering, in 2015. Her doctoral research focused on spatio-temporal routing under various constraints specific to multi-robot applications. She received an MS degree in Electrical and Computer Engineering from Georgia Tech, in 2012, and a BTech degree in Electronics and Communications from the Manipal Institute of Technology, India, in 2009.

Giuseppe Notarstefano Giuseppe Notarstefano is Associate Professor at the Universit del Salento (Lecce, Italy), where he was Assistant Professor (Ricercatore) from February 2007 to May 2016. He received the Laurea degree summa cum laude in Electronics Engineering from the Universit di Pisa in 2003 and the Ph.D. degree in Automation and Operation Research from the Universit di Padova in 2007. He has been visiting scholar at the University of Stuttgart, University of California Santa Barbara and University of Colorado Boulder. His research interests include distributed optimization, cooperative control in complex networks, applied nonlinear optimal control, and trajectory optimization and maneuvering of aerial and car vehicles. He serves as an Associate Editor for IEEE Transactions on Control Systems Technology, for IEEE Control Systems Letters, for the Conference Editorial Board of the IEEE Control Systems Society and for other IEEE and IFAC conferences. He coordinated the VIRTUS team winning the International Student Competition Virtual Formula 2012. He is recipient of an ERC Starting Grant 2014.

Matthew Rice Matthew Rice recently graduated from the Georgia Institute of Technology. He received his Master's degree in Electrical and Computer Engineering, where his research involved applications of spatio-temporal routing in the context of multi-robot systems. He received his BS in Electrical Engineering from Georgia Tech in 2013.

Magnus Egerstedt Magnus Egerstedt is the Executive Director for the Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology and a Professor and the Julian T. Hightower Chair in Systems and Controls in the School of Electrical and Computer Engineering. He received the M.S. degree in Engineering Physics and the Ph.D. degree in Applied Mathematics from the Royal Institute of Technology, Stockholm, Sweden, the B.A. degree in Philosophy from Stockholm University, and was a Postdoctoral Scholar at Harvard University. Dr. Egerstedt is a Fellow of the IEEE and a recipient of a number of research and teaching awards, including the Ragazzini Award from the American Automatic Control Council.