Distributed Algorithm Design for Multi-Robot Task Assignment with Deadlines for Tasks

Lingzhi Luo, *Student Member, IEEE*, Nilanjan Chakraborty, *Member, IEEE*, and Katia Sycara, *Fellow, IEEE*

***Abstract*— In this paper, we present provably-good algorithms for multi-robot task assignment, where each task has to be completed within its deadline. Each robot has a upper limit on the maximum number of tasks that it can perform due to its limited battery life, and each task takes the same amount of time to complete. Each robot has a different payoff (or cost) for the tasks and the objective is to assign the tasks to the robots such that the total payoff (cost) is maximized (minimized) while respecting the task deadline constraints. This problem is an extension of a special generalized assignment problem (where each task consumes the same time resource and must be finished), with additional deadline constraints for the time resource assignment. We show that the problem can be reduced to a problem of assigning tasks to robots, where the tasks are organized in overlapping sets, and each robot has a limit on the number of tasks it can perform from each set, which is a variant of multi-robot assignment problem with set precedence constraint (*SPC-MAP*) discussed in [1]. We present a distributed auction-based algorithm for this problem and prove that the solution is almost-optimal. We also present simulation results to depict the performance of our algorithm.**

I. INTRODUCTION

Multi-robot task assignment is a fundamental problem that arises in a wide variety of application scenarios like manufacturing, automated transport of goods, environmental monitoring and surveillance. In some application scenarios the tasks have to be completed within given deadlines. Furthermore, the assignment should be good in the sense that it should maximize a payoff function or minimize a cost function. Assigning tasks to robots to meet the deadline constraints as well as maximize the overall payoff of assignment is a type of scheduling problem. Scheduling is a quite mature field and due to its importance in a wide variety of application areas including manufacturing and computer systems different types of scheduling problems have been studied [2]. The problem in this paper is related to deterministic offline scheduling problems with resource constraints [3] (in contrast to online and/or stochastic scheduling problems). Although batch scheduling has been well studied, most scheduling algorithms are centralized in nature and usually there is no limit on the number of jobs that a processor or machine can perform. For multi-robot application scenarios, energy of the robots is a key constraint and so the number of tasks that a robot can do in any mission is bounded. Furthermore, distributed algorithms that enable the robots in the field to divide the tasks among themselves (so that there is no central point of failure) is desirable. Thus, in this paper, our goal is to design distributed algorithms for task allocation with task deadlines and capacity limits on the total number of tasks a robot can perform.

The authors are with the Robotics Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, {lingzhil, nilanjan, katia}@cs.cmu.edu

Task allocation with deadlines is relevant for many multi-robot application scenarios. Consider the situation where a system of robots have to clear up objects from one area and place them in other areas. This can arise in automated package handling in ports where packages have to be unloaded from a container (e.g., a ship) and placed in other containers (e.g., trucks). Furthermore, there may be a deadline on the tasks coming from the need to complete the overall task of unloading within a certain time. Another application area is for removing debris in disaster recovery scenario where the robots need to move objects from one place to another so that the paths become usable by other robots that have to reach potential victims. In such cases also there may be a deadline for the robots to clear paths because victims should be found and reached within some time. In these applications, different robots might have heterogeneous capacities, which have different fitness for different tasks, so the objective here could be maximizing the quantitative fitness of robot-task assignment while respecting task deadlines.

The general problem that we consider in this paper is as follows: *We are given a set of tasks T, with each task $t_j \in T$ having a deadline d_j. Each task has to be done by one robot only and each robot can do one task at a time. The maximum number of tasks that robot r_i can do is N_i (this is called the budget of the robot). Each robot r_i obtains a payoff a_{ij} for doing task t_j. The overall payoff is the sum of the individual robot payoffs. The objective is to assign the tasks to robots such that the deadline constraints are met and the overall payoff is maximized*. We assume that each task takes unit duration. Note that we could have equivalently stated the problem above in terms of cost minimization. When we leave task deadlines unspecified, the problem becomes a linear assignment problem, which can be solved using the Hungarian algorithm [4], [5], [6], parallel auction algorithm [7], [8], or distributed auction-based algorithm [9], [10]. When we further allow tasks to have different processing time, the problem becomes the NP-hard generalized assignment problem, where approximation algorithms exist [11], [12]. So our problem is an extension of the linear assignment problem, a special generalized assignment problem, with added feature of task deadline constraints.

We present a distributed auction-based algorithm, where each robot can bid for its own task, and show that this algorithm provides an *almost optimal* solution. We first show that the deadline constraints provide a natural grouping of the

tasks into overlapping sets and the problem can be equivalently formulated as a problem of assigning tasks to robots such that there is an upper bound on the number of tasks that can be performed from each set. This is a natural variant of the multi-robot assignment problem with set precedence constraints ($SPC-MAP$) in [1], where tasks are forming disjoint groups and each robot can perform at most one task from each group. We show that solving this problem can be reduced to solving a min-cost network flow problem and hence our problem can be solved in polynomial time. Further, we present an auction-based distributed algorithm that provides an almost-optimal solution (i.e., a solution that is within $O(n_t\varepsilon)$, where n_t is the total number of tasks and ε is a parameter to be chosen). By appropriately choosing ε, we can make our solution arbitrarily close to the optimal solution (however at the cost of more computation time). We also present simulation results showing the performance of our algorithm for understanding the effect of choice of ε.

II. Related Work

Task allocation is important in many applications of multi-robot systems, e.g., multi-robot routing [13], multi-robot decision making [14], and other multi-robot coordination problems (see [15], [10]). There are different variations of the multi-robot assignment problem that have been studied in the literature depending on the assumptions about the tasks and the robots (see [16], [15], [17] for surveys), and there also exists multi-robot task allocation systems (e.g., Traderbot [18], [19], Hoplites [20], MURDOCH [21], ALLIANCE [22]) that build on different algorithms. In this paper, we consider a deterministic offline multi-robot assignment with task deadline constraints, and our objective is to design distributed algorithms with provable performance guarantee. Therefore, we will restrict our discussion to most relevant literature with performance guarantee.

In the simplest version of the task allocation problem (also known as the linear assignment problem), each robot can perform at most one task and the robots are to be assigned to tasks such that the overall payoff is maximized. The linear assignment problem is essentially a maximum weighted matching problem for bipartite graphs, which can be solved in a centralized manner using the Hungarian algorithm [4], [6], or a decentralized manner with shared memory using auction algorithm [23], or a totally distributed way using consensus-based auction algorithm [9], [10]. However all of this work assume that the tasks are independent, and there does not exist such constraint as deadlines on tasks. In [1], set precedence constraints are introduced among tasks, where the tasks are organized into disjoint groups such that each robot can be assigned to at most one task from each group and there is a bound on the number of tasks that a robot can do. A generalization of the auction algorithm of [23] is presented in [1] to achieve an almost optimal solution. Our problem extends the problem in [1] in the sense that the task group can overlap, and each robot can be assigned to multiple tasks in each group.

Assigning tasks with deadlines to parallel machines have been studied in scheduling literature [2]. However, the common objective there is either to find a feasible solution so that task deadlines are met [3], or to minimize the weight of unscheduled late jobs [24], instead of maximizing the total payoff of different machine-task matching (this feature is a departure of our work from the standard scheduling problems studied in the literature).

III. Problem Formulation

In this section, we give the formal definition of our multi-robot assignment problem with deadlines for independent tasks with identical duration ($DiMAP$). Here an assignment is not just to determine which robot performs which tasks, but also to make sure that the robot performs the tasks in proper time, i.e., any task is assigned to a certain time slot of one robot' schedule so that its deadline constraint is satisfied. Since the assignments would be related to the processing time of tasks, we first give the straightforward formulation of the problem using a time-related parameter, and then derive an equivalent formulation, which removes the time parameter and facilitates the algorithm design in Section IV.

Suppose that there are n_r robots, $R=\{r_1,\ldots,r_{n_r}\}$, and n_t tasks, $T=\{t_1,\ldots,t_{n_t}\}$ where the tasks are independent, and each task t_j has a unit duration with a deadline d_j, define $D=\max_j d_j$ as the maximum task deadline, $S_k=\{t_j|d_j=k\},\forall k=1,\ldots,D$, as the set of tasks with deadline k, $S_{D+1}=\{t_j|d_j \text{ is not specified}\}$ as tasks with no explicit deadline; each robot r_i has N_i available time slots in its schedule, and thus r_i can perform at most N_i tasks, i.e., robot r_i's *budget* is N_i. Any robot can be assigned to any task, and performing each task needs a single robot, so $n_t \le \sum_{i=1}^{n_r} N_i$. Let f_{ij}^k be the variable that takes a value 1 if task, t_j, is assigned to the k-th time slot of robot, r_i, and 0 otherwise, where $i\in\{1,\ldots,n_r\}, j\in\{1,\ldots,n_t\}, k\in\{1,\ldots,N_i\}$. Let $a_{ij}\in\mathbb{R}$ be the payoff for the assignment pair (r_i,t_j), i.e., for assigning robot r_i to task t_j, which does not depend on the assigned time slot k. The objective is to assign all tasks to robots so that the total payoffs from the assignment is maximized while the deadlines of tasks are satisfied. The problem can be formulated as an integer linear program (ILP) below.

$$\max_{\{f_{ij}^k\}} \quad \sum_{i=1}^{n_r}\sum_{j=1}^{n_t}\sum_{k=1}^{N_i} a_{ij} f_{ij}^k$$

$$\text{s.t.} \quad \sum_{i=1}^{n_r}\sum_{k=1}^{N_i} f_{ij}^k = 1,\ \forall j=1,\ldots,n_t \tag{1}$$

$$\sum_{i=1}^{n_r}\sum_{k=1}^{\min(N_i,d_j)} f_{ij}^k = 1,\ \forall j=1,\ldots,n_t \tag{2}$$

$$\sum_{j=1}^{n_t} f_{ij}^k \le 1,\ \forall i=1,\ldots,n_r, k=1,\ldots,N_i \tag{3}$$

$$f_{ij}^k \in \{0,1\},\ \forall i,j,k \tag{4}$$

where (1) means that each task is assigned to exactly one time unit of a robot's schedule; (2) guarantees that each task is assigned to a time slot before its deadline; (3) guarantees

that each time slot of robots is assigned to at most one task and thus each robot r_i is assigned to at most N_i tasks.

The problem formulation above adds a time-related parameter k so that the deadline constraints for tasks can easily be represented in (2), and its solution will also give a fixed schedule that specifies, which robots perform which tasks during each time step. However, it might be unnecessary in terms of maximizing the total payoffs, e.g., it does not matter whether robot r_i perform task t_j at time step k_1 or k_2 (assuming both satisfy the task deadline d_j) since both would lead to the same payoff a_{ij}. Below we provide another equivalent problem formulation, showing that we can explore the independency of tasks so that explicit time parameter k can be removed while all constraints are still satisfied. Let f_{ij} be the variable that takes a value 1 if task, t_j, is assigned to robot, r_i, and 0 otherwise. The problem can be formulated as an integer linear program (ILP) given below.

$$\max_{\{f_{ij}\}} \quad \sum_{i=1}^{n_r}\sum_{j=1}^{n_t} a_{ij}f_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_r} f_{ij} = 1,\ \forall j = 1,\ldots,n_t \tag{5}$$

$$\sum_{j:d_j\le l} f_{ij} \le l,\ \forall i = 1,\ldots,n_r, l = 1,\ldots,D \tag{6}$$

$$\sum_{j=1}^{n_t} f_{ij} \le N_i,\ \forall i = 1,\ldots,n_r \tag{7}$$

$$f_{ij} \in \{0,1\},\ \forall i,j \tag{8}$$

where (5), corresponding to (1), means that each task is assigned to exactly one time slot of one robot's schedule; (6), corresponding to (2), guarantees that each robot is assigned to at most l tasks from all tasks with deadline no more than l, and thus each task can be performed before its deadline; (7), corresponding to (3), guarantees that each robot r_i does not exceed its budget, i.e., is assigned to at most N_i tasks.

The solution for the second problem formulation only determines which robot performs which tasks without explicitly modeling the assignment of each time slot of robots to tasks. However, due to (6), for all tasks with deadlines no more than l, each robot can be assigned to l of them. Thus it can be guaranteed that such assignment would satisfy the deadline constraints for tasks since the number of tasks assigned to each robot is no more than the available time slots of that robot before the tasks' deadlines. The second problem formulation is more compact since it does not explicitly use the time parameter as in the first problem formulation. In next section, we do not consider the processing time for tasks anymore, and show that this formulation is convenient for our distributed algorithm design. The linear program (LP) relaxation of *DiMAP* can be reduced to a min-cost network flow problem, so there is always an optimal integer solution for its LP relaxation. Due to space constraint, we leave the reduction in the future complete version.

IV. Algorithm Design and Performance Analysis

In this section, we show that the distributed auction algorithm used for multi-robot assignment with set precedence constraints ($SPC-MAP$) in [1] can be extended to get an almost-optimal solution for *DiMAP*.

A. Basic Idea and Concepts of Auction Algorithm

We are trying to match n_r robots and n_t tasks with constraints (5)-(8) through a market auction mechanism as introduced in [23], where each robot is an economic agent acting in its own best interest. Although each robot r_i wants to be assigned to its favorite N_i tasks (with highest payoffs) while satisfying the deadline constraints for tasks, the different interest of robots will probably cause conflicts. This can be resolved by introducing auxiliary variables of task price, and making robots bid for tasks through an iterative auction mechanism. Suppose the price for task t_j at iteration τ is $p_j(\tau)$, so the net value of task t_j to robot r_i at iteration τ becomes $v_j(\tau) = a_{ij} - p_j(\tau)$ instead of just a_{ij}. During the bidding procedure, each robot bids for tasks which satisfy the constraints and have highest values to the robot according to certain rule (as shown later in Section IV-B). After winning the bids and assigned to tasks in each iteration, the robot would then set the new task price as the winning bid, which is the highest bid value for the task among all robots till then. Thus the iterative bidding from robots leads to the evolution of robot-task assignment as well as task price $p_j(\tau)$, which can gradually resolve the interest conflicts among robots. [1]

Please note, since $\sum_i N_i \ge n_t$, we need add $\sum_i N_i - n_t$ virtual tasks with small equal payoffs to all robots, and leave their deadlines as unspecified. So the new total number of tasks becomes $n_t' = \sum_i N_i$. Besides, the condition that each robot must know the current price $p_j(\tau)$ for all task t_j during bidding procedure requires the existence of a centralized auctioneer or a shared memory for all robots to access. In [1], [9], [10], maximum consensus technique has been introduced to combine with auction algorithm so that the algorithm becomes totally distributed without centralized auctioneer to communicate the current price of tasks with robots. Assume that robots are forming a connected communication network, where each robot is connected to its neighboring robots within its communication range. The idea is that during each bidding iteration τ, each robot r_i in the connected network locally maintains and updates a list of current highest bids $p_j^i(\tau)$ [2] for each tasks t_j from its own neighborhood $\mathcal{N}_{r_i}$:

$$p_j^i(\tau) = \max_{r_\ell \in \mathcal{N}_{r_i}} p_j^\ell(\tau - 1)$$

and uses that highest bid as local price of tasks. Since the network is connected, the global highest bids would eventually propagate to all robots so that the solution quality remains the same as that of original auction algorithm. The

[1] Note that $p_j(\tau)$ is an auxiliary variable, which is used to resolve the conflict that multiple robots share the same interest of being assigned to the same tasks. When the algorithm terminates, the quality of assignment solution does not depend on $p_j(\tau)$, i.e., the output assignment solution is almost-optimal in terms of original payoffs a_{ij} instead of the net value $v_j(\tau) = a_{ij} - p_j(\tau)$.

[2] Note that each robot just maintains one price for each task, here $p_j^i(\tau)$ is just used to represent the task price at iteration τ for convenience.

same technique is applied here to make our new auction-based algorithm totally distributed.

Below we will discuss some important concepts of auction algorithm. Suppose $\mathscr{T}_{J_i} = \{t_j | j \in J_i\}$ is the task set assigned to robot r_i, it must satisfy the constraints below:

$$|J_i| \leq N_i, |\mathscr{T}_{J_i} \bigcap (\bigcup_{m=1}^{k} S_m)| \leq k, \forall k = 1, \ldots, D \tag{9}$$

where $|J_i| \leq N_i$ corresponds to constraint (7), $|\mathscr{T}_{J_i} \bigcap (\bigcup_{m=1}^{k} S_m)| \leq k, \forall k = 1, \ldots, D$ corresponds to (6), and the exclusive assignment would guarantee (5). We use $J_i \sim (9)$ to represent that J_i satisfies (9).

During each bidding iteration τ, given any task price set $\{p_j(\tau) | j = 1, \ldots, n_t\}$, every robot r_i wants to be exclusively assigned to a task set $\mathscr{T}_{J_i^*} = \{t_j | j \in J_i^*\}$ with maximum net values while satisfying the constraints:

$$J_i^* = \arg \max_{\forall J_i \sim (9)} \sum_{j \in J_i} v_j(\tau) \tag{10}$$

We say robot r_i is *happy* with the assigned task set $\mathscr{T}_{J_i^*}$ when (10) is satisfied. If all robots are happy, we say the whole assignment and the prices at iteration τ are *at equilibrium*.

Suppose we fix a positive scalar ε. When each assigned task for robot r_i is within ε of being in the set of r_i's maximum values, that is,

$$f(J_i') \geq \max_{\forall J_i \sim (9)} \sum_{j \in J_i} (v_j(\tau) - \varepsilon) \tag{11}$$

where $f(J_i') = \sum_{j \in J_i'} v_j(\tau)$ and $J_i' \sim (9)$. We say robot r_i is *almost happy* with the assigned task set $\mathscr{T}_{J_i'}$ when (11) is satisfied. If all robots are almost happy, we say the whole assignment and the prices at iteration τ are *almost at equilibrium*.

B. Auction-based Distributed Algorithm Design

In this section, we design a new auction-based distributed algorithm for *DiMAP*, which is an extension of the algorithm used in [1] for multi-robot assignment problem with set precedence constraints (*SPC-MAP*). In the distributed algorithm, there is no centralized component, and the knowledge/information available to each robot r_i is $\{a_{ij} | \forall j\}$, the payoffs of tasks to r_i itself, as well as $\{p_j^{\ell}(\tau) | \forall r_\ell \in \mathscr{N}_{r_i} \cup \{r_i\}, \forall j, t\}$, the local task price maintained and updated in each neighboring robot r_ℓ during each bidding iteration τ.

For each robot r_i, a single bidding iteration τ of our auction-based algorithm is described in Algorithm 1. Each robot could implement the iterative bidding procedure either synchronously or asynchronously. For the sake of ease of discussion, below we assume that in our auction-based algorithm, all robots run copies of Algorithm 1 sequentially. Each bidding iteration τ for robot r_i (Algorithm 1) can be summarized as follows.

First, robot r_i communicates with its neighbors to get their maintained local task price, updates its own local task price (from Line 2 to 5), and computes each task value to itself (Line 7). Please note, the updated local task price at robot r_i is a local maximum of each task price among its neighborhood (including itself), which is a lower bound of the real task price. The real task price is the global highest bid value among all robots, and can be achieved by maximizing the local task price maintained by all robots.

Second, given the current local task price $\{p_j^i(\tau) | \forall j\}$, robot r_i selects a task set with task indices J, so that it is *happy* to be assigned to the task set $\mathscr{T}_J = \{t_j | j \in J\}$, i.e., (10) is satisfied, (from Line 11 to 18 inside the iterative loop). This part guarantees that all constraints for robot r_i are satisfied (according to the value of k' from Line 12 to 16): (a) robot r_i is assigned to at most N_i tasks; (b) r_i is assigned to at most k tasks of all tasks with deadline no more than k. Meanwhile each task is assigned to at most one robot, because each task either does not change assignment status (assigned to previous robot or remains unassigned) or switch from the previous assigned robot to robot r_i.

Third, robot r_i is assigned to task set $\mathscr{T}_J$, and updates the task price (from Line 29 to 32) so that $\forall j \in J, p_j^i(\tau+1) = p_j^i(\tau) + (v_j^i(\tau) - v_{J^{\triangle}(j)}^i(\tau)) + \varepsilon$, where $v_{J^{\triangle}(j)}^i(\tau)$ is the value of the task $J^{\triangle}(j)$, which would have been selected to J had we removed task j. For each assigned task in $t_j \in \mathscr{T}_J$, there is a corresponding task $J^{\triangle}(j)$, which is stored in $J^{\triangle}$ indexed by j (Line 20 explains how $J^{\triangle}$ is computed from Line 19 to 27). Roughly speaking, $J^{\triangle}(j)$ is the task with the second value to r_i other than j while satisfying the constraints together with other tasks in $\mathscr{T}_{J \setminus \{j\}}$. The bidding price for each task is at least ε bigger than its previous price:
$p_j^i(\tau+1) - p_j^i(\tau) = v_j^i(\tau) - v_{J^{\triangle}(j)}^i(\tau) + \varepsilon \geq \varepsilon$
since the selection of $J^{\triangle}$ from Line 19 to 27 guarantees that $v_{J^{\triangle}(j)}(\tau) \leq v_j(\tau)$. So the tasks receiving r_i's bids must be assigned to r_i at the end of the iteration. The way we set $p_j^i(\tau+1)$ guarantees that r_i is *almost happy* with $\mathscr{T}_J$ given the new price $p_j^i(\tau+1)$ (See Theorem 2), and is related to the proof of the optimality of the algorithm, which will be discussed in Section IV-C.

The algorithm terminates when all robots have been exclusively assigned to their own tasks. Each robot needs to wait until its task price information does not change for n_t rounds, which is the largest possible diameter of any connect network with n_t nodes. In this way, each robot can make sure the unchanged task price is not due to the delay of price propagation in the network, and can terminate the algorithm in a distributed way.

Here the feasibility check and adding virtual tasks could be easily done before each robot runs Algorithm 1. However, it is possible to integrate these steps into each robot's distributed bidding procedure. Due to the space constraint, we will leave these issues in the future complete version.

C. Performance Analysis

In this section, we analyze the performance of Algorithm 1 in terms of soundness, completeness and optimality, i.e., does the output assignment solution satisfy all constraints in (5)-(8)? Will Algorithm 1 terminate with a feasible assignment solution in a finite number of iterations? How good is the

Algorithm 1 Auction Iteration τ For Robot r_i

Input: a_{ij}, $p_j^{\ell}(\tau)$, S_k *for all* $j,k,r_\ell \in \mathcal{N}_{r_i} \cup \{r_i\}$,
Output: $p_j^i(\tau+1)$, J *// J: indices of* r_i*'s assigned tasks*
// Update the local highest bid information:
for each task t_j **do**
  $p_j^i(\tau) = \max_{r_\ell \in \mathcal{N}_{r_i} \cup \{r_i\}} p_j^{\ell}(\tau)$
end for
// Collect information for new bids
Denote $v_j^i(\tau) = a_{ij} - p_j^i(\tau)$ *// value of* t_j *to* r_i
$J = \emptyset$, $J^{\triangle} = zeros(n_t, 1)$
// Iterate over tasks with different deadlines k
for $k = 1 : D+1$ **do**
  $J' = J \bigcup S_k$; *// J' :current candidate task set*
  if $k \leq D$ **then**
    $k' = \min(k, N_i)$; *// k' :number of tasks to be selected*
  else
    $k' = \min(|J'|, N_i)$;
  end if
  Select the best k' candidate tasks from J', and store their indices into J:
  $J = \arg(\max^{(k')})_{j \in J'} v_j^i(\tau)$ *// $\arg(\max^{(k')})$ is the operator to get indices of the k' biggest values*
  Store the index of next best candidate task from J': $j' = \arg\max_{j \in J' \setminus J} v_j^i(\tau)$
  For each selected task in J, update the index of its corresponding next candidate task (with highest value among all next best candidate tasks since the iteration when it is first selected) into $J^{\triangle}$:
  for each task t_a: $a \in J$ **do**
    if $a \in S_k$ **then**
      $J^{\triangle}(a) = j'$; *// for task with deadline k*
    else
      $J^{\triangle}(a) = \arg\max(v_{j'}^i(\tau), v_{J^{\triangle}(a)}^i(\tau))$; *// $d_a < k$*
    end if
  end for
end for
// Start new bids and update price information
Bid with price b_j for task $t_j : j \in J$:
$b_j = p_j^i(\tau) + v_j^i(\tau) - v_{J^{\triangle}(j)}^i(\tau) + \varepsilon$, $p_j^i(\tau+1) = b_j$;
for task $t_j : j \notin J$, $p_j^i(\tau+1) = p_j^i(\tau)$

solution when Algorithm 1 terminates?

Lemma 1: When Algorithm 1 terminates for all robots, the achieved assignment must be a feasible solution for *DiMAP*, i.e., (5)-(8) are satisfied.

Proof: When Algorithm 1 for robot r_i terminates, according to the value of k', (a) r_i has already been assigned to no more than N_i tasks and no other robot would bid higher for r_i's assigned tasks; (b) r_i is assigned to at most l tasks of all tasks with deadline no more than l. So (6) and (7) are satisfied. Since the tasks are exclusively assigned in Algorithm 1, (5) and (8) are also satisfied. So the achieved assignment is a feasible solution satisfying (5)-(8).■

Lemma 1 means Algorithm 1 is sound, i.e., when it outputs a solution, the solution is feasible. The next result asserts that Algorithm 1 always terminates in finite number of iterations assuming the existence of at least one feasible assignment for the problem. The proof relies on the observations below:

(a) When a task is assigned, it will remain assigned during the whole process of the algorithm. The reason is: during the bidding and assignment process, one task can either transfer from unassigned to assigned, or be reassigned from one robot to another, but cannot become unassigned from assigned. There might exist cases where one task was assigned to more than one robot before the algorithm terminates due to the local price information.

(b) Each time when a task receives a bid, its new price will increase by at least ε according to the algorithm. So if one task receives infinite number of bids, its price will become $+\infty$. Please note, although the real task price might not reach all robots immediately, the $+\infty$ price would eventually propagate to all robots.

(c) If a robot r_i bids for infinite number of times, at least one task t_j would receive infinite number of bids. Suppose that $t_j \in S_k$, then all tasks in $S = S_k \bigcup S_{k+1} \bigcup \ldots \bigcup S_{D+1}$ would receive infinite number of bids. The reason is that: (using contradiction) if there exists one task in S, which does not receive infinite number of bids, its price would be finite, and its value for r_i must be bigger than t_j which receives infinite number of bids. So it has to receive more bids, which leads to the contradiction. So all tasks in S receive infinite number of bids and thus have the price of $+\infty$ (according to (b)).

Theorem 1: If there is at least one feasible solution for an instance of *DiMAP*, Algorithm 1 for all robots will terminate in a finite number of iterations.

Proof: If the algorithm continues infinitely, there must exist a smallest k_0, s.t. all tasks in $S = S_{k_0} \bigcup S_{k_0+1} \bigcup \ldots \bigcup S_{D+1}$ have $+\infty$ price according to (c) above. For each robot r_i, either $N_i < k_0$, in this case, robot r_i is assigned to tasks in $T \setminus S$; or $N_i \geq k_0$, in this case, r_i must be assigned to exactly $k_0 - 1$ tasks in $T \setminus S$ since all $k_0 - 1$ tasks selected in the procedure of Algorithm 1 must have larger value than tasks in S. So the remaining number of unassigned tasks for all robots are $\sum_{N_i \geq k_0} (N_i - k_0 + 1)$. Since all tasks in S have $+\infty$ price, they must keep the assigned status although they might be assigned to more than one robot and their assigned robots keep changing according to (a), so

$$\sum_{i:N_i \geq k_0} (N_i - k_0 + 1) > |S|$$

Please note that the above inequality is strict, since there must be at least one robot r_i with $N_i \geq k_0$ that has remaining tasks unassigned (otherwise no robot would continue to bid, and the algorithm would terminate). Since $\sum_i N_i = n_t'$(including the additional virtual tasks),

$$\sum_{i:N_i < k_0} N_i + \sum_{i:N_i \geq k_0} (k_0 - 1) < n_t' - |S|$$

where the left part of the inequality represents the maximum number of tasks, which all robots can perform within deadline $k_0 - 1$, while the right part represents the number of

tasks with deadline smaller than k_0. So the inequality means that there exist at least one task with deadline smaller than k which cannot be performed within its deadline, so there is no feasible solution for the instance of *DiMAP*, which leads to the contradiction. So we conclude that Algorithm 1 must terminate in a finite number of iterations if there exists a feasible solution for an instance of *DiMAP*. ■

Lemma 1 and Theorem 1 together prove that Algorithm 1 is both sound and complete. Next we want to prove the performance of Algorithm 1, based on the following theorem.

Theorem 2: After each iteration τ of robot r_i, r_i's newly assigned tasks together with the local task prices $p_j^i(\tau+1)$ keep r_i *almost happy*, i.e., (11) is satisfied.

Proof. During each iteration τ, according to the bidding part of Algorithm 1 (from Line 11 to 18), the bid tasks $\mathscr{T}_J = \{t_j | j \in J\}$ with the price before the iteration can make r_i *happy*:

$$f(J) = \sum_{j\in J}(a_{ij} - p_j^i(\tau)) = \max_{\forall J_i \sim (9)} \sum_{j\in J_i}(a_{ij} - p_j^i(\tau))$$

$p_j^i(\tau+1) = p_j^i(\tau) + v_j^i(\tau) - v_{J^{\triangle}(j)}^i(\tau) + \varepsilon, \forall j \in J$, and $p_j^i(\tau+1) = p_j^i(\tau), \forall j \notin J$, so

$$\begin{aligned} f'(J) &= \sum_{j\in J}(a_{ij} - p_j^i(\tau+1)) = \sum_{j\in J}(v_{J^{\triangle}(j)}^i(\tau) - \varepsilon) \\ &= \max_{\forall J_i \sim (9)} \sum_{j\in J_i}(a_{ij} - p_j^i(\tau+1) - \varepsilon) \end{aligned}$$

So after each iteration τ, the values of tasks in J make robot r_i *almost happy*, which means (11) is satisfied. ■

Since Theorem 2 holds true for all robots, we get the corollary below.

Corollary 1: When Algorithm 1 for all robots terminates, the achieved assignment and price are *almost at equilibrium*.

Theorem 3 below analyzes the optimality and gives performance guarantee for Algorithm 1.

Theorem 3: When Algorithm 1 for all robots terminates, the achieved assignment $\{(i, J_i^*) | i = 1, \ldots, n_r\}$ must be within $\sum_{i=1}^{n_r} N_i \varepsilon$ of an optimal solution.

Proof: Denote $(\{(i, J_i) | i = 1, \ldots, n_r\}$ as any feasible assignment, i.e., $\{J_i | \forall i\} \sim (12)$:

$$|J_i \bigcap (\bigcup_{n=1}^{m} S_n)| \leq m, \forall i, m : i = 1, \ldots, n_r; m = 1, \ldots, D$$

$$J_i \bigcap J_j = \emptyset \text{ if } i \neq j, \quad |J_i| \leq N_i, \forall i, \quad |\bigcup_{i=1}^{n_r} J_i| = n_t$$

Denote $\{p_j^* | j = 1, \ldots, n_t\}$ as the set of task prices when Algorithm 1 terminates for all robots and $\{p_j | j = 1, \ldots, n_t\}$ as any set of task prices.

First, we give an upper bound for the optimal solution.

$$\sum_{i=1}^{n_r}\sum_{j\in J_i}(a_{ij} - p_j) \leq \max_{\forall \{J_i' | \forall i\} \sim (12)} \sum_{i=1}^{n_r}\sum_{j\in J_i'}(a_{ij} - p_j)$$

$$\Rightarrow \sum_{i=1}^{n_r}\sum_{j\in J_i} a_{ij} \leq \sum_{j=1}^{n_t} p_j + \max_{\forall \{J_i' | \forall i\} \sim (12)} \sum_{i=1}^{n_r}\sum_{j\in J_i'}(a_{ij} - p_j)$$

Since it holds true for any set of price $\{p_j | \forall j\}$ and any feasible assignment $\{(i, J_i) | \forall i\}$, we have $A^* \leq D^*$, where A^* is the optimal total payoffs of any feasible assignment.

$$A^* = \max_{\forall \{J_i | \forall i\} \sim (12)} \sum_{i=1}^{n_r}\sum_{j\in J_i} a_{ij}$$

$$D^* = \min_{p_j; j=1,\ldots,n_t} (\sum_{j=1}^{n_t} p_j + \max_{\forall \{J_i' | \forall i\} \sim (12)} \sum_{i=1}^{n_r}\sum_{j\in J_i'}(a_{ij} - p_j))$$

On the other hand, according to Corollary 1, we have

$$\sum_{i=1}^{n_r}\sum_{j\in J_i^*}(a_{ij} - p_j^*) \geq \max_{\forall \{J_i' | \forall i\} \sim (12)} \sum_{i=1}^{n_r}\sum_{j\in J_i'}(a_{ij} - p_j^* - \varepsilon)$$

$$\begin{aligned} \sum_{i=1}^{n_r}\sum_{j\in J_i^*} a_{ij} &\geq \sum_{j=1}^{n_t} p_j^* + \max_{\forall \{J_i' | \forall i\} \sim (12)} \sum_{i=1}^{n_r}\sum_{j\in J_i'}(a_{ij} - p_j^*) - \sum_{i=1}^{n_r} N_i \varepsilon \\ &\geq D^* - \sum_{i=1}^{n_r} N_i \varepsilon \geq A^* - \sum_{i=1}^{n_r} N_i \varepsilon \end{aligned}$$

$\sum_{i=1}^{n_r}\sum_{j\in J_i^*} a_{ij}$ is the total payoffs of the achieved assignment by Algorithm 1, and

$$A^* \geq \sum_{i=1}^{n_r}\sum_{j\in J_i^*} a_{ij} \geq A^* - \sum_{i=1}^{n_r} N_i \varepsilon$$

So it is within $\sum_{i=1}^{n_r} N_i \varepsilon$ of an optimal solution.■

V. Simulation Results

In this section, we run simulations in a synthetic example to check how the control parameter ε influences the auction algorithm's solution quality and convergence time.

Consider $n_r = 20$ robots, each robot r_i needs to perform $N_i = 5$ tasks from $n_t = 100$ tasks. The deadlines of tasks are randomly set so that there are 15 tasks for each deadline from 1 to 5, respectively, and 10 tasks without deadline.

ε is a control parameter related to the convergence time and performance guarantee of Algorithm 1. In our simulations, we tested different values of ε. For each ε, we generated 100 rounds of random payoffs a_{ij} from a uniform distribution in $(0, 20)$, and we compared the mean and standard deviation of performance ratio of our solution to the optimal solution, and the convergence time of the algorithm.

Figure 1 shows how the solution of assignment payoffs changes with the control parameter ε. When ε is as small as 0.1, the assignment payoffs achieved by our algorithm almost equal the optimal solution. When ε increases, the difference between our solution and the optimal solution is increased, but bounded by $\sum_{i=1}^{n_r} N_i \varepsilon$, as proven in Theorem 3. Figure 2 shows how the convergence time of our algorithm changes with ε. The number of rounds[3] decreases with ε, which means with higher ε, Algorithm 1 converges faster.

From Figure 1 and 2, we can see that, similar as results in [1], there is a tradeoff between the solution quality and the convergence time, which can be adjusted by ε. With bigger ε,

[3]One round is defined as the procedure that all robots finish one bidding iteration sequentially.

the algorithm converges faster at sacrifice of solution quality; while with smaller ε, the algorithm solution is better at the cost of slower convergence time. In our simulation, robots are forming fully connected network. If robot network is not fully connected, the convergence time would be delayed depending on the diameter of the network.

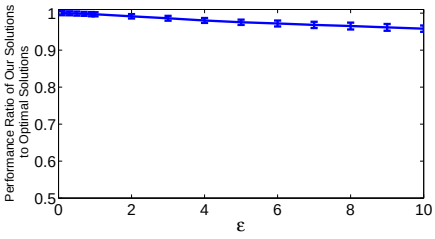


Fig. 1. Performance ratio of our solution to the optimal one as a function of ε, which is the minimum price increase for new bids. The optimal solution can be achieved when we set $\varepsilon < \frac{min_diff}{\sum_{i=1}^{n_r} N_i}$ where min_diff is the minimum difference between any two individual payoffs a_{ij}.

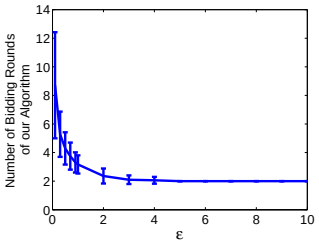


Fig. 2. Convergence time of our algorithm as a function of ε. The figure shows the number of rounds for our algorithm to terminate, where one round means all robots sequentially implement Algorithm 1 for one iteration.

VI. Summary

In this paper we considered the multi-robot task assignment problem with task deadline constraints (*DiMAP*), where the objective is to maximize the total payoff of assigning tasks to robots while respecting the task deadline constraint and robot budget constraint. This problem can be reformulated so that tasks are organized in overlapping groups according to their deadlines, and each robot has a limit on the number of tasks it can perform in each group. We presented a distributed auction-based algorithm, and proved that our algorithm are sound, complete and almost-optimal.

Future work: One natural extension would be to consider the case when tasks have different durations, which is an extension of the NP-hard *generalized assignment problem* (with added feature of deadline constraints). To solve this problem *in a distributed way* each robot during each iteration solves a single-robot NP-hard problem (an extension of *knapsack problem*). We plan to explore if designing an approximation algorithm for the single robot problem will lead to an approximate solution for the overall problem.

Acknowledgments

This work was partially supported by AFOSR MURI grant FA95500810356, ONR grant N000140910680, and NSF award IIS-1218542.

References

[1] L. Luo, N. Chakraborty, and K. Sycara, "Multi-robot assignment algorithms for tasks with set precedence constraints," in *Proceedings of ICRA*, May 2011.

[2] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[3] J. Blazewicz, "Solving the resource constrained deadline scheduling problem via reduction to the network flow problem," *European Journal of Operational Research*, vol. 6, no. 1, pp. 75 – 79, 1981.

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

[5] A. V. Goldberg, E. Tardos, and R. E. Tarjan, *Paths, Flows and VLSI-Design (eds. B. Korte, L. Lovasz, H.J. Proemel, and A. Schrijver)*. Springer Verlag, 2009, ch. Network Flow Algorithms, pp. 101–164.

[6] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Society for Industrial and Applied Mathematics, 2009.

[7] D. P. Bertsekas and D. A. Castanon, "The auction algorithm for transportation problems," *Annals of Operations Research*, vol. 20, pp. 67–96, 1989.

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

[9] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *Proc. 47th IEEE Conf. Decision and Control*, 2008, pp. 1212–1217.

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

[11] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, pp. 162–166, 2006.

[12] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *Proc. of ACM-SIAM SODA*, 2006, pp. 611–620.

[13] 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.

[14] C. Bererton, G. Gordon, S. Thrun, and P. Khosla, "Auction mechanism design for multi-robot coordination," in *NIPS*, 2003.

[15] 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, jul. 2006.

[16] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[17] A. R. Mosteo and L. Montano, "A survey of multi-robot task allocation," Instituto de Investigacin en Ingenierła de Aragn (I3A), Tech. Rep., 2010.

[18] A. Stentz and M. B. Dias, "A free market architecture for coordinating multiple robots," CMU Robotics Institute, Tech. Rep., 1999.

[19] M. B. Dias and A. Stentz, "A free market architecture for distributed control of a multirobot system," in *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, July 2000, pp. 115 – 122.

[20] N. Kalra, D. Ferguson, and A. Stentz, "Hoplites: A market-based framework for planned tight coordination in multirobot teams," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 1170 – 1177.

[21] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics*, vol. 18, no. 5, pp. 758–768, October 2002.

[22] L. Parker, "Alliance: an architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220 –240, apr 1998.

[23] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, pp. 105–123, 1988.

[24] D. Karger, C. Stein, and J. Wein, "Algorithms and theory of computation handbook," M. J. Atallah and M. Blanton, Eds., 1997, ch. Scheduling Algorithms.