Distributed Algorithm Design for Multi-robot Generalized Task Assignment Problem

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

***Abstract*— We present a provably-good distributed algorithm for generalized task assignment problem in the context of multi-robot systems, where robots cooperate to complete a set of given tasks. In multi-robot generalized assignment problem (MR-GAP), each robot has its own resource constraint (e.g., energy constraint), and needs to consume a certain amount of resource to obtain a payoff for each task. The objective is to find a maximum payoff assignment of tasks to robots such that each task is assigned to at most one robot while respecting robots' resource constraints. MR-GAP is a NP-hard problem. It is an extension of multi-robot linear assignment problem since different robots can use different amount of resource for doing a task (due to the heterogeneity of robots and tasks). We first present an auction-based iterative algorithm for MR-GAP assuming the presence of a shared memory (or centralized auctioneer), where each robot uses a knapsack algorithm as a subroutine to iteratively maximize its own objective (using a modified payoff function based on an auxiliary variable, called price of a task). Our iterative algorithm can be viewed as (an approximation of) best response assignment update rule of each robot to the assignment of other robots at that iteration. We prove that our algorithm converges to an assignment (approximately) at equilibrium under the assignment update rule, with an approximation ratio of $1+\alpha$ (where α is the approximation ratio for the Knapsack problem). We also combine our algorithm with a message passing mechanism to remove the requirement of a shared memory and make our algorithm totally distributed assuming the robots' communication network is connected. Finally, we present simulation results to depict our algorithm's performance.**

I. INTRODUCTION

Task assignment is a fundamental problem in multi-robot system with various applications such as intelligent manufacturing, automated transport of goods, search and rescue assistance in disaster relief, as well as environmental monitoring. In the basic formulation of multi-robot linear assignment problem, it is assumed that each task would consume the same unit amount of resource from each robot's resource budget. However, in practice, each task might consume different amount of resource from different robots due to the heterogeneity of robots and tasks, which can be modeled as multi-robot generalized assignment problem (MR-GAP). In MR-GAP, each robot has its own resource constraint, and needs to consume a certain amount of resource to obtain a payoff for each task. The overall objective is to find a maximum payoff assignment of tasks to robots such that each task is assigned to at most one robot while respecting robots' resource budget constraints. Given its wide applicability for real-world problems in various areas as well as its computational NP-hardness, generalized assignment problem (GAP) has been well studied in operations research, theoretical computer science and other related research communities. However, most algorithms are centralized in nature. In multi-robot application scenarios where robots need to autonomously operate in the field, it is desirable to have distributed algorithms on individual robots so that the system is resilient to single-point failure and adaptive to environmental change. Thus, in this paper, our goal is to design distributed algorithms for MR-GAP with provable performance guarantee.

Multi-robot generalized task assignment arises in many multi-robot application scenarios. Especially when tasks and robots are heterogeneous, the amount of resource each task consume from each robot, as well as the payoff each robot could obtain from each task, might be different. Depending on the specific application, the resource could be energy, processing time or any other consumable resource. Consider the situation in automated warehouse management system where packages have to be picked up from certain clustered storage locations, and placed in other delivery locations. In this situation, different robots and objects might be distributed across different spatially clustered location. Thus, the energy each robot consume to travel from its original position to the targeted object location could be different. Another application area is in disaster recovery scenario where the robots need to remove debris and clear the paths. In such cases different robots with heterogeneous design might need different processing time to remove different kinds of debris.

In this paper, we present a distributed auction-based algorithm for MR-GAP, where each robot can bid for its own tasks by solving a knapsack sub-problem as subroutine. We show that our algorithm provides an $1+\alpha$ approximate solution assuming that the knapsack problem is solved by an algorithm with approximation ratio $\alpha \in [1, +\infty)$. Thus, our distributed algorithm has an approximation ratio of 2 (or 3), when the algorithm used for knapsack is optimal (or 2-approximate). Unlike other approximation algorithms of GAP, our auction-based new algorithm is designed specifically for distributed multi-robot systems with limited range communication. Furthermore, our algorithm can achieve a similar approximation ratio with a competitive running time. Our proof also presents a new perspective showing that best-response assignment update rule of individual robots would lead to an assignment at equilibrium with guaranteed approximation ratio. We first present our auction-based iterative algorithm for MR-GAP assuming that the robots have access to a shared memory (or there is a centralized auctioneer).

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

Each robot obtains the information of highest bid for each task among all robots from the shared memory, and then uses a knapsack algorithm as a subroutine to iteratively maximize its own objective (using a modified payoff function based on an auxiliary variable called price of a task). The assignment update rule of our iterative algorithm can be viewed as (approximate)[1] best response of each robot to the temporary assignment of other robots at that iteration. We prove that our algorithm would eventually converge to an assignment at (approximate) equilibrium with an approximation ratio of $1+\alpha$. We also make our algorithm totally distributed by combining it with a message passing mechanism to remove the requirement of a shared memory (at the cost of slower convergence and more local communication), assuming the robots' communication network is connected. Finally, we present simulation results to depict the performance of our algorithm.

II. Related Work

Task allocation is important in many applications of multi-robot systems, e.g., multi-robot routing [1], multi-robot decision making [2], and other multi-robot coordination problems (see [3], [4]). 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 [5], [3], [6] for surveys), and there also exists multi-robot task allocation systems (e.g., Traderbot [7], [8], Hoplites [9], MURDOCH [10], ALLIANCE [11]) that build on different algorithms. Here we consider a deterministic offline multi-robot generalized assignment problem, 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 [12], [13], or a decentralized manner with shared memory using auction algorithm [14], or a totally distributed way using consensus-based auction algorithm [15], [4]. However all of this work assume that the tasks are independent. Some work has been done to address the constraints among tasks in multi-robot task assignment. In [16], 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 [14] is presented in [16] to achieve an almost optimal solution. [17] studied the multi-robot task assignment with task deadline constraints, which extends the problem in [16] in the sense that the task group can overlap, and each robot can be assigned to multiple tasks in each group. The constrained linear assignment problems in [16], [17] are solvable in polynomial time whereas MR-GAP is NP-hard.

Generalized assignment problem (GAP) is an extension to the linear assignment problem, which has been extensively studied in both operation research [18], [19] and theoretical computer science [20], [21], [22], [23]. However, most algorithms are centralized in nature, i.e., a centralized controller collects all parameter information and then computes the whole assignment. This may not be suitable for situations where distributed algorithm is required for multi-robot in-field operation. A branch and bound algorithm was presented in [18] to determine the bounds of optimal solution. A series of 0/1 knapsack problem are solved so that the bound gets refined iteratively. A branch-and-price algorithm was designed in [19] that employs both column generation and branch-and-bound to obtain optimal integer solutions. However, these algorithms do not provide any approximation guarantee. Some approximation algorithms exist for GAP, e.g., LP-based 2-approximation algorithm in [20], [21]. A combinatorial local search with $(2+\varepsilon)$-approximation guarantee, and an LP-based algorithm with $(\frac{e}{e+1}+\varepsilon)$-approximation guarantee with polynomial running time are presented in [23]. A $(2+\varepsilon)$-approximation algorithm with the same guarantee as the combinatorial local search but a better running time is given in [22]. The algorithm presented in [22] can be viewed as the first round of our iterative algorithm where each robot sequentially runs the algorithm for one iteration.

III. Problem Formulation

Suppose that there are n_r robots, $R=\{r_1,\ldots,r_{n_r}\}$, and n_t tasks, $T=\{t_1,\ldots,t_{n_t}\}$. Each robot, r_i, has resource budget N_i, and consumes resource w_{ij} to complete task t_j while getting payoff a_{ij}. Any robot can be assigned to any task, and performing each task needs a single robot. The objective is to assign tasks to robots so that the sum of the payoffs of the robots is maximized subject to the resource constraints. Let f_{ij} take a value 1 if task t_j is assigned to robot r_i and 0 otherwise, where $i\in\{1,\ldots,n_r\}$, $j\in\{1,\ldots,n_t\}$. We study the maximization version of MR-GAP, which can be formulated as an integer linear program (ILP):

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

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

$$\sum_{j=1}^{n_t} w_{ij} f_{ij} \le N_i, \ \forall i=1,\ldots,n_r \tag{2}$$

$$f_{ij} \in \{0,1\}, \ \forall i,j \tag{3}$$

where (1) guarantees that each task is exclusively assigned to at most one robot; (2) guarantees that the sum of consumed resources for tasks assigned to each robot r_i does not exceed its budget N_i. When $w_{ij}=1$ and $N_i=1$, the generalized assignment problem becomes the linear assignment problem [13]. When $w_{ij}=w_j$ and $a_{ij}=a_j$, i.e., w_{ij} and a_{ij} do not

[1]Approximate best response and at approximate equilibrium will be strictly defined in Definition 3 and 4.

vary for different robots, the generalized assignment problem becomes a multiple knapsack problem [24].

IV. Algorithm Design and Performance Analysis

In this section, we introduce an iterative auction-based algorithm for multi-robot generalized assignment problem. We will first introduce a few key concepts such as robot's (approximate) best response and the assignment at (approximate) equilibrium. We also recall the definition of knapsack problem. We will then present an iterative auction-based algorithm with shared memory, where given current temporary assignment of other robots, each robot bids for tasks using the knapsack algorithm as a subroutine. We show the connection of our algorithm to (approximate) best response update rule, and prove that the algorithm would converge to an assignment at (approximate) equilibrium with guaranteed approximation ratio. Finally, we discuss the use of a message passing mechanism to make our algorithm totally distributed.

A. Preliminary Concepts

Let $J_i = \{j | f_{ij} = 1\}$ denote the task set assigned to robot r_i and $J = \cup_i \{J_i\}$ be a task assignment solution for GAP.

Definition 1: Define an *assignment transform function* G_i as a transformation from a given old assignment J' to a new assignment J, due to a new assignment component J_i for robot r_i: $J = G_i(J', J_i) = (\cup_{k \neq i}\{J'_k \setminus J_i\}) \cup \{J_i\}$, i.e.,

$$J_k = \begin{cases} J_i & \text{if } k = i \\ J'_k \setminus J_i & \text{if } k \neq i \end{cases}$$

We say J_i is a feasible assignment for robot r_i if and only if J_i satisfies r_i's budget constraint in (2), denoted as $J_i \sim (2)$; and J is a feasible assignment, if and only if J satisfies all constraints in (1) - (2), denoted as $J \sim (1) - (2)$.

Lemma 1: The assignment transform function G_i is a valid transform, i.e., if both J' and J_i are feasible assignments, then $J = G_i(J', J_i)$ is also feasible.

Proof: For any robot $r_k \neq r_i$, its newly assigned task set $J_k = J'_k \setminus J_i \subset J'_k$. Since J'_k is feasible for r_k, J_k must also be feasible, i.e., the subset of previously assigned tasks must consume less resource than the budget of r_k. Besides, $J_k \cap J_i = \emptyset$, so J must exclusively assign tasks to at most one robot. Together with the feasibility of J_i, we know that the new assignment $J \sim (1) - (2)$, i.e., the transform function is valid. ■

Denote $F(J) = \sum_{i:J_i \in J} \sum_{j \in J_i} a_{ij}$ as the total payoff of a feasible assignment J; $H(J', J_i) = F(G_i(J', J_i)) - F(G_i(J', \emptyset))$ as the total payoff increment due to a new assignment component of robot r_i from $\emptyset$ to J_i, imposed on J'.

Definition 2: A new assignment component J_i^* is robot r_i's *best response*[2] to an old assignment J' if and only if

$$J_i^* = \arg\max_{J_i} H(J', J_i)$$

which is the best unilateral assignment change of robot r_i to increase the total payoff from assigning nothing to r_i.

[2]Note that r_i's best response might not always be unique for some given old assignment J'. In such cases, we could use any one as the best response.

Definition 3: A new assignment component J_i^* is robot r_i's α*-approximate best response* to an old assignment J' ($\alpha \in [1, +\infty)$), if and only if

$$\alpha H(J', J_i^*) \geq \max_{J_i} H(J', J_i)$$

Definition 4: An assignment J^* is *at equilibrium* (or *at* α*-approximate equilibrium*) if and only if any assignment component $J_i^* \in J^*$ is already robot r_i's best response (or α-approximate best response) to J^* itself, i.e.,

$$\forall J_i^* \in J^* : J_i^* = \arg\max\nolimits_{J_i} H(J^*, J_i)$$
$$(\text{or } \alpha H(J^*, J_i^*) \geq \max\nolimits_{J_i} H(J^*, J_i))$$

Note that if we use (α-approximate) best response as the iterative assignment update rule for each robot, any assignment at (α-approximate) equilibrium would be a fixed point for such update rule. There might be many different assignments at (α-approximate) equilibrium depending on the parameters of problem instances.

Since we use algorithms for 0/1 knapsack problem as a subroutine in our iterative algorithm later, we recall the definition of 0/1 knapsack problem below.

Definition 5: [**0/1 Knapsack Problem**]: Consider n items, $\{x_1, \ldots, x_n\}$, and a bag to contain these items. Each x_i has a value v_i and weight w_i. The maximum weight that we can carry in the bag is W. Assume that all values and weights are nonnegative. The objective is to determine the items of maximum value such that the total weight is less than or equal to W.

$$\max_{\{y_i \in \{0,1\}\}} \sum_{i=1}^{n} v_i y_i \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i y_i \leq W.$$

where $y_i = 1$ if item x_i is in the bag, otherwise $y_i = 0$. The knapsack optimization problem is NP-hard. There exist a pseudo-polynomial time algorithm using dynamical programming and a fully polynomial time approximation scheme (FPTAS). The FPTAS uses the pseudo-polynomial algorithm as a subroutine, and can approximate the optimal solution to any specified degree in polynomial time [24].

B. Auction-based Decentralized Algorithm Design

We want to match n_r robots and n_t tasks with constraints (1)-(3) through a market auction mechanism, where each robot is an economic agent acting in its own best interest to bid for tasks. Each robot r_i wants to be assigned to its favorite tasks (with highest payoffs) while satisfying its budget constraints in (2). The different interest of robots will probably cause conflicts in assignment that violate the constraints in (1). This can be resolved by introducing auxiliary variables called task price, and making robots bid for tasks with highest values (defined as payoffs minus price) instead of highest payoffs, through an iterative auction mechanism.

At iteration τ, let the price for task t_j be $p_j(\tau)$. The value of task t_j to robot r_i is $v_{ij}(\tau) = a_{ij} - p_j(\tau)$ instead of just a_{ij}. Robot r_i bids for tasks which satisfy its budget constraints and have highest values to itself. Formally, in iteration τ, robot r_i computes its new bids by solving the following $0/1$

knapsack problem:

$$\max_{\{f_{ij}\in\{0,1\}\}} \sum_{j=1}^{n_t} v_{ij}(\tau) f_{ij} \quad \text{s.t.} \quad \sum_{j=1}^{n_t} w_{ij} f_{ij} \le N_i. \tag{4}$$

Let J_i be the task set obtained by robot r_i by solving the problem (4) using an α-approximation algorithm for the knapsack problem. Robot r_i would then bid for each task t_j, $j \in J_i$, with new price a_{ij}, which would guarantee r_i to win the bids since $v_{ij}(\tau) = a_{ij} - p_j(\tau) > 0$. We assume that there exists a shared memory (or auctioneer) for all robots to access the current task price, which is the current highest bid from all robots. The shared memory is also used to guarantee that at any time, at most one robot can access the task price and provide new bids for tasks. After winning the bids and assigned to tasks in the iteration, the robot would then set the new task price as the winning bid, which is the highest bid 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. [3]

Based on the idea described above, we design a new auction-based decentralized algorithm for the generalized assignment problem. In the decentralized algorithm, there is no centralized controller to make assignment decisions for robots. Instead each robot are making assignment decision by itself. 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. However, the shared memory must guarantee that at any time, at most one robot can access the task price and provide new bids for tasks. For the sake of ease of discussion, below we assume that in our auction-based algorithm, all robots run copies of Algorithm 1 sequentially. The algorithm terminates after the task price information does not change after all robots bid for one iteration.

As shown in Algorithm 1 (Line 1), the knowledge / information available to each robot r_i during its bidding iteration τ includes two parts: (a) locally maintained information: $\{a_{ij}|\forall j\}$ and $\{w_{ij}|\forall j\}$, the payoffs of tasks to r_i itself and their consumed resource for r_i, J_i' and $\{b_j'|j \in J_i'\}$, indices of tasks assigned to r_i during its previous bidding iteration and r_i's bidding price for those tasks at that iteration; (b) information accessed from the shared memory: $\{p_j(\tau)|\forall j\}$, the task price maintained and updated in the shared memory during its bidding iteration τ.

First, robot r_i goes through tasks in J_i', which is the task set assigned to r_i during its previous bidding iteration. r_i compares the current price of those tasks with the corresponding previous bids b_j' from r_i: if $b_j' < p_j(\tau)$, it means that another robot must have bid higher price for t_j, and thus t_j has been reassigned to the robot with that bid; otherwise, $b_j' = p_j(\tau)$, task t_j is still assigned to robot r_i since b_j' is still the highest bid. In the latter case, r_i resets the task price to be zero so that the new value of the task to r_i is still a_{ij}. (Line 2 to 8)

Second, given the current task price $\{p_j(\tau)|\forall j\}$, robot r_i selects a task set with task indices J_i^* using any knapsack algorithm with performance guarantee to maximize the total assignment values $\sum_{j\in J_i^*} v_{ij}(\tau)$ (Line 9 to 11).

Third, robot r_i is assigned to task set J_i^*, and updates the task price (from Line 12 to 15) so that $\forall j \in J_i^*, p_j(\tau+1) = a_{ij}$. The bidding price for each task is a_{ij} bigger than its previous price $p_j(\tau)$ (otherwise $v_{ij}(\tau) = a_{ij} - p_j(\tau) \le 0$, t_j would not be selected), so the tasks receiving r_i's bids must be assigned to r_i at the end of the iteration.

Algorithm 1 Auction Iteration τ For Robot r_i

1: *Input:* a_{ij}, $p_j(\tau)$, $\forall j$, J_i', $\{b_j'|j \in J_i'\}$ // *J_i': indices of r_i's previously assigned tasks*
Output: $p_j(\tau+1)$, J_i^* // *J_i^*: r_i's newly assigned tasks*
2: // *Reset the price of still assigned tasks from previous iteration to zero*
3: **for** each task t_j: $j \in J_i'$ **do**
4:   **if** $p_j(\tau) == b_j'$ **then**
5:     $p_j(\tau) = 0$;
6:     $p_j(\tau+1) = 0$;
7:   **end if**
8: **end for**
9: // *Collect information for new bids*
10: Denote $v_{ij}(\tau) = a_{ij} - p_j(\tau)$ // *value of t_j to r_i*
11: $J_i^* = knapsack(v_{ij}(\tau), w_{ij}, N_i)$;
12: // *Start new bids and update price information*
13: Bid with price b_j for task $t_j : j \in J_i^*$:
14: $b_j = a_{ij}$, $p_j(\tau+1) = b_j$;
15: for task $t_j : j \notin J_i^*$, $p_j(\tau+1) = p_j(\tau)$

C. Performance Analysis

In this section, first, we show the connection of Algorithm 1 to robot's (approximate) best response update rule; second, we prove that the algorithm would converge to an assignment at (approximate) equilibrium; third, we prove that the assignment at (α-approximate) equilibrium is guaranteed to be a solution for GAP with approximation ratio $1+\alpha$. Below we assume that the subroutine knapsack algorithm in Algorithm 1 has $\alpha \in [1, +\infty)$ approximation ratio[4].

Lemma 2: When robot r_i runs Algorithm 1 at iteration τ, its newly assigned task set J_i^* is $\alpha - approximate$ *best response* to the assignment at the beginning of iteration τ.

Proof: Suppose the assignment at the beginning of iteration τ is J'. $\forall$ a new feasible assignment J_i for robot r_i, the total value increment due to J_i would be

$$\begin{aligned} H(J', J_i) &= F(G_i(J', J_i)) - F(G_i(J', \emptyset)) \\ &= \sum_{k \ne i}\Big(\sum_{j \in J_k'} a_{kj} - \sum_{j \in J_i \cap J_k'} a_{kj}\Big) + \sum_{j \in J_i} a_{ij} - F(G_i(J', \emptyset)) \end{aligned}$$

[3]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 evaluated in terms of original payoffs a_{ij} instead of the net value $v_{ij}(\tau) = a_{ij} - p_j(\tau)$.

[4]Note that there exists pseudo-polynomial time algorithm to achieve optimal solution for knapsack problem. In that case, $\alpha = 1$

$$= \sum_{k\neq i}\sum_{j\in J'_k} a_{kj} + \sum_{j\in J_i}(a_{ij} - p_j(\tau)) - F(G_i(J',\emptyset))$$
$$= \sum_{j\in J_i}(a_{ij} - p_j(\tau))$$

which is the objective of knapsack problem, solved by r_i as a subroutine in Algorithm 1. Since we assume that the knapsack algorithm leads to $\alpha-$approximate solution,

$$\alpha \sum_{j\in J_i^*}(a_{ij} - p_j(\tau)) \geq \max_{J_i\sim(2)} \sum_{j\in J_i}(a_{ij} - p_j(\tau)) \Rightarrow$$
$$\alpha H(J', J_i^*) \geq \max_{J_i\sim(2)} H(J', J_i)$$

According to Definition 3, we get that J_i^* is $\alpha - approximate$ *best response* to J' at the beginning of iteration τ. ■

Theorem 1: Algorithm 1 for all robots will terminate in a finite number of iterations, and converges to an assignment at $\alpha - approximate\ equilibrium$.
Proof: When $\alpha = 1$, according to Lemma 2, it is easy to see that the new assignment J_i^* for robot r_i would make the total assignment payoff non-decreasing. In the case that $\alpha > 1$, we could easily incorporate a simple comparison in the knapsack routine so that the output would be the better of J'_i and J_i^*, and thus the new total assignment payoff is still non-decreasing with each iteration of new bids. Besides, the total payoff is bounded. So Algorithm 1 for all robots will terminate in a finite number of iterations.

When Algorithm 1 for all robots terminates, according to Lemma 2 and Definition 4, it must converge to an assignment at $\alpha - approximate\ equilibrium$. ■
When $\alpha = 1$, Algorithm 1 is actually r_i's *best response*, and it would converge to an assignment *at equilibrium*. According to the proof above, the convergence time of Algorithm 1 would be $O(n_r \cdot f(n_t) \cdot C)$ where $f(n_t)$ is the running time for knapsack algorithm and C is a constant due to the number of iterations, depending on the payoff parameters(i.e., the maximum total payoff divided by the minimum payoff increment).

Theorem 2: An assignment at $\alpha - approximate\ equilibrium$ is a solution for GAP with approximation ratio $1+\alpha$.
Proof: Suppose the assignment at $\alpha - approximate\ equilibrium$ is $J^* = \cup_i\{J_i^*\}$, while the optimal assignment is $J^{opt} = \cup_i\{J_i^{opt}\}$. Below we want to compare the total payoff of each robot r_i in two different assignment J_i^* and J_i^{opt}. Since J_i^* must be α-approximate best response to J^*,

$$\alpha \sum_{j\in J_i^*}(a_{ij} - p_j) \geq \sum_{j\in J_i^{opt}}(a_{ij} - p_j) \tag{5}$$

There are two cases depending on whether $\bar{J_i} = J_i^{opt} \cap (\cup_{k\neq i} J_k^*) = \emptyset$ or not:

(a) If $\bar{J_i} = \emptyset$: According to Algorithm 1, $\forall j \notin \cup_i J_i^*, p_j = 0$,

$$\sum_{j\in J_i^*} p_j \geq \sum_{j\in J_i^{opt}} p_j \tag{6}$$

Combining Equation (5) and (6) above, we have that

$$\alpha \sum_{j\in J_i^*} a_{ij} \geq \sum_{j\in J_i^{opt}} a_{ij} \tag{7}$$

TABLE I
PAYOFF PARAMETERS a_{ij} AND CONSUMED RESOURCE PARAMETERS w_{ij} IN EXAMPLE 1

a_{ij}	t_1	t_2
r_1	1	$\alpha+\varepsilon$
r_2	$1+\alpha\varepsilon$	ε

w_{ij}	t_1	t_2
r_1	1	1
r_2	1	1

If $\forall i \in \{1,\ldots,n_r\}$, $\bar{J_i} = \emptyset$, we have

$$\alpha \sum_i \sum_{j\in J_i^*} a_{ij} \geq \sum_i \sum_{j\in J_i^{opt}} a_{ij} \tag{8}$$

So J^* is a solution with approximation ratio α.

(b) If $\bar{J_i} \neq \emptyset$: again since $\forall j \notin \cup_i J_i^*, p_j = 0$,

$$\sum_{j\in J_i^*} p_j \geq \sum_{j\in J_i^{opt}\setminus\bar{J_i}} p_j = \sum_{j\in J_i^{opt}} p_j - \sum_{j\in\bar{J_i}} p_j \tag{9}$$

Combining Equation (9) and (5), we have that

$$\alpha \sum_{j\in J_i^*} a_{ij} + \sum_{j\in\bar{J_i}} p_j \geq \sum_{j\in J_i^{opt}} a_{ij} \tag{10}$$

If $\forall i \in \{1,\ldots,n_r\}$, $\bar{J_i} \neq \emptyset$, we have

$$\alpha \sum_i \sum_{j\in J_i^*} a_{ij} + \sum_i \sum_{j\in\bar{J_i}} p_j \geq \sum_i \sum_{j\in J_i^{opt}} a_{ij} \tag{11}$$

Since $\forall i_1, i_2$, $J_{i_1}^{opt} \cap J_{i_2}^{opt} = \emptyset \Rightarrow \bar{J}_{i_1} \cap \bar{J}_{i_2} = \emptyset$. So
$\sum_i \sum_{j\in\bar{J_i}} p_j \leq \sum_i \sum_{j\in J_i^*} p_j = \sum_i \sum_{j\in J_i^*} a_{ij}$
Together with Equation (11),

$$(\alpha+1) \sum_i \sum_{j\in J_i^*} a_{ij} \geq \sum_i \sum_{j\in J_i^{opt}} a_{ij} \tag{12}$$

So J^* is a solution with approximation ratio $1+\alpha$.
Since $\forall i$, either $\bar{J_i} = \emptyset$ or $\bar{J_i} \neq \emptyset$, it must belong to one of the two cases above. So it is guaranteed that the assignment J at $\alpha - approximate\ equilibrium$ is a solution for GAP with approximation ratio $\max(\alpha, 1+\alpha) = 1+\alpha$. ■
According to Theorem 1 and 2, we prove that Algorithm 1 would eventually converge to a solution for GAP with approximation ratio $1+\alpha$. The following example shows that the approximation ratio of assignments at $\alpha - approximate$ equilibrium is actually tight.

Example 1: Consider two robots with budget $N_1 = N_2 = 1$, and two tasks, with parameters listed in Table I, where ε is an arbitrarily small constant. The assignment $\{J_1 = \{t_1\}, J_2 = \{t_2\}\}$ is an assignment at $\alpha - approximate$ equilibrium:

$$\begin{aligned} & \alpha(F(G_{i_1}(J,J_1)) - F(G_{i_1}(J,\emptyset))) = \alpha((1+\varepsilon)-\varepsilon) \\ \geq\ & (\alpha+\varepsilon) - \varepsilon = F(G_{i_1}(J, J_1^* = \{t_2\})) - F(G_{i_1}(J,\emptyset)); \\ & \alpha(F(G_{i_2}(J,J_2)) - F(G_{i_2}(J,\emptyset))) = \alpha((1+\varepsilon)-1) \\ \geq\ & (1+\alpha\varepsilon) - 1 = F(G_{i_2}(J, J_2^* = \{t_1\})) - F(G_{i_2}(J,\emptyset)) \end{aligned}$$

However, it is an $(1+\alpha)$ approximate solution to the optimal assignment $\{J_1^* = \{t_2\}, J_2^* = \{t_1\}\}$:

$$(1+\alpha)F(J) = (1+\alpha)(1+\varepsilon) = ((\alpha+\varepsilon)+(1+\alpha\varepsilon)) = F(J^*)$$

D. Distributed Implementation

Algorithm 1 is decentralized in the sense that every robot can make assignment decisions by itself, based on an iteratively updated common information of task price from the shared memory. In this section, we discuss how to remove the requirement of the existence of shared memory to make the algorithm totally distributed assuming the robots' communication network is connected.

Suppose that there exists a robot communication network $G=(V,E)$, where $V=R$ consists of robot nodes, and $E=\{(i_1,i_2)\}$ consists of connection edges between robots, which can directly communicate. We assume that G is connected.

In a distributed implementation of Algorithm 1, no shared memory exists to provide task price $p_j(\tau)$ during each iteration τ. Each robot r_i needs to locally maintain the task price $p^i_j(\tau)$, and update them based on the local communication with its direct neighbor in $\mathcal{N}_i=\{i'|(i',i)\in E\}$.

Below, we show that a distributed message passing mechanism could be used for robot to maintain and update the task price information in a distributed way. During each iteration τ, robot r_i runs Algorithm 1, where $p_j(\tau)$ would become the local maintained task price $p^i_j(\tau)$, to get the new assignment J_i and new task price $p^i_j(\tau+1)$. The message passing mechanism is described as follows.

First, r_i would send out the message in the following format: $M_i^{\tau+1}=(P,r_i,V,\tau+1)$, where $P=(p^i_1(\tau+1),\ldots,p^i_{n_t}(\tau+1))$ is the new price vector for all tasks maintained in r_i, r_i is the identifier of the robot who sends out the message, $V=\sum_{j\in J_i}v_{ij}(\tau)$ is the output total value of the knapsack subroutine algorithm in Algorithm 1, and $\tau+1$ is time stamp of the message, i.e., the number of iteration when the message would be used to update the task price. If $J_i=J'_i$, i.e., the robots' bidding tasks are the same as before, V is set to be 0 in P.

Second, when r_i receives a message $M_{i'}^{\tau+1}$ from one of its neighbor i_0, it would first send out the message to its neighbors except i_0. Then r_i would compare $M_{i'}^{\tau+1}(V)$ with its locally maintained $V_{\max}(\tau+1)$, which is the maximum value of all messages with time stamp $\tau+1$ till then. If $M_{i'}^{\tau+1}(V)>V_{\max}(\tau+1)$, r_i would store the message with higher value and reset $V_{\max}(\tau+1)=M_{i'}^{\tau+1}(V)$, and get rid of previous message; if $M_{i'}^{\tau+1}(V)<V_{\max}(\tau+1)$, r_i would get rid of the message $M_{i'}^{\tau+1}$. To break the tie when $M_{i'}^{\tau+1}(V)=V_{\max}(\tau+1)$, robots could use a consistent rule, e.g., keep the message with the smaller robot identifier.

Third, r_i would keep track of the number of robot identifiers $n_{ID}(\tau+1)$ from all messages. When $n_{ID}(\tau+1)=n_r$, i.e., r_i has received all robots' messages for iteration $\tau+1$, r_i would start to update its locally maintained task price from the only stored message (e.g., $M_{i'}^{\tau+1}$) with the highest value: $p^i_j(\tau+1)=M_{i'}^{\tau+1}(P(j)),\forall j$, and then start a new bidding procedure for iteration $\tau+1$.

From the above message passing mechanism, we know that during each iteration τ, each robot would start a new bid and send out a new message. Since the robot communication network G is connected, all messages would reach all robots. However, only the message with highest value from $r^*(\tau)$ would be stored and used to update task price for $\tau+1$, which would be consistent among all robots. It is equivalent to say that during each iteration τ, only one robot $r^*(\tau)$ starts a new bid, and updates task price, which would be consistently and locally stored by all robots. Thus we can see that although the shared memory is removed, its two following functions are still maintained in a distributed way: (a) during any iteration, at most one robot can start a new bid and update task price; (b) task price are consistently maintained among all robots. So the conclusions in Section IV-C are valid in the distributed implementation. However, since the bidding message needs to be propagated in the network G, during each iteration, the distributed algorithm might be delayed by the product of one-hop message passing time and Δ ($\Delta\le n_r$), which is the diameter of G.

V. Simulation Results

In this section, we present some preliminary simulation results to check how our algorithm's solution quality changes with iterations till convergence. Consider $n_r=20$ robots, where each robot r_i has budget $N_i=10$, and $n_t=40$ tasks. In our simulations, we first assume each robot can communicate with all other robots, i.e., $\Delta=1$. The knapsack algorithm used in the simulation is the optimal dynamic programming algorithm, so $\alpha=1$ and the approximation ratio of Algorithm 1 is 2.

Figure 1 and Figure 2 show that in two different simulation samples how the solution performance changes with bidding iterations of robots. In both figures, we randomly generate 100 samples with different a_{ij} and w_{ij}, and show the mean and standard deviation of our solution performance. In all the 100 generated samples, our algorithm converges within 200 iterations. In Figure 1, for each robot r_i and task t_j, payoffs a_{ij} are drawn from a uniform distribution in $(0,9)$, and the consumed resource w_{ij} from $[1,6]$. In Figure 2, for each robot r_i and task t_j, we set the consumed resource $w_{ij}=5,\forall i,j$, and a_{ij} are randomly generated according to the distributions in Table II, where $U(x_{min},x_{max})$ represents a uniform distribution from x_{min} to x_{max}. From Figure 2 and Figure 1, we can see that although the total assignment payoffs get improved until convergence in both cases, the improvement patterns before convergence are very different in the two cases: in Figure 1, the assignment performance after all robots run one iteration is very close to the performance of assignment at convergence, while Figure 2 shows that in some situations, our algorithm could achieve much better solution than the algorithm where all robots run one iteration. The reason is that when all robots just run one iteration, robots bidding first might lose their assigned tasks to robots bidding later, and do not have chance to be assigned to other tasks, which could be compensated in our iterative algorithm.

VI. Summary

We studied the multi-robot generalized assignment problem, where the objective is to maximize the total assignment payoffs while respecting robots' budget constraints. We

TABLE II

PAYOFF PARAMETERS a_{ij} DISTRIBUTIONS IN FIGURE 2

a_{ij}	t_1 - t_{20}	t_{21} - t_{40}
r_1 - r_{10}	$U(8,9)$	U(6,7)
r_{11} - r_{20}	$U(10,11)$	U(0,1)

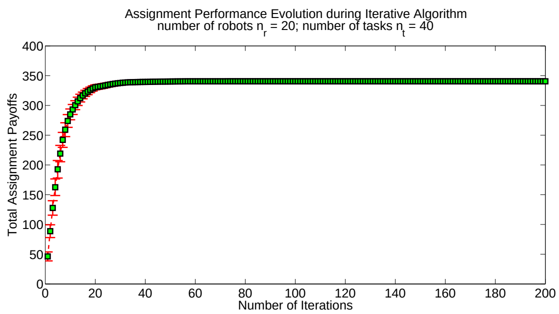

Fig. 1. Statistics of total assignment payoffs by our algorithm as a function of iterations, where a_{ij} and w_{ij} are randomly generated in 100 samples.

presented a distributed auction-based algorithm, where each robot iteratively uses a knapsack algorithm as subroutine to choose its assigned tasks and maximize the sum of each assigned task value (defined as a task's payoff minus its price). Suppose the knapsack subroutine algorithm has an approximation ratio $\alpha \in [1, +\infty)$. We show that the iterative bidding procedure of each robot is actually an α-approximate best response assignment update rule to the current temporary assignment of other robots. We proved that such bidding procedure would eventually converge to an assignment at α-approximate equilibrium, which is guaranteed to be a solution to MR-GAP with an approximation ratio of $1+\alpha$. We also presented simulation results illustrating our algorithm.

ACKNOWLEDGMENTS

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

REFERENCES

[1] 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.

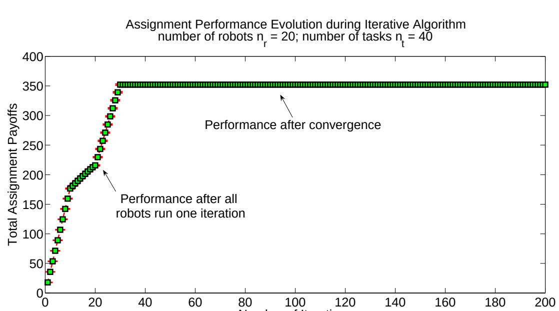

Fig. 2. Statistics of total assignment payoffs as a function of iterations, where parameters w_{ij} are randomly generated while a_{ij} are carefully designed according to distributions in Table II.

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

[3] 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.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

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

[15] 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.

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

[17] ——, "Distributed algorithm design for multi-robot task assignment with deadlines for tasks," in *Proceedings of IEEE International Conference on Robotics and Automation, 2013*, May 2013.

[18] G. Ross and R. Soland, "A branch and bound algorithm for the generalized assignment problem," *Mathematical Programming*, vol. 8, pp. 91–103, 1975.

[19] M. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem," *Operations Research*, vol. 45, pp. 831–841, 1997.

[20] D. B. Shmoys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, no. 3, pp. 461–474, Dec. 1993.

[21] C. Chekuri and S. Khanna, "A ptas for the multiple knapsack problem," in *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '00, 2000, pp. 213–222.

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

[23] 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.

[24] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.