Competitive Analysis of Repeated Greedy Auction Algorithm for Online Multi-Robot Task Assignment

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

***Abstract*— We study an online task assignment problem for multi-robot systems where robots can do multiple tasks during their mission and the tasks arrive dynamically in groups. Each robot can do at most one task from a group and the total number of tasks a robot can do is bounded by its limited battery life. There is a payoff for assigning each robot to a task and the objective is to maximize the total payoff. A special case, where each group has one task and each robot can do one task is the online maximum weighted bipartite matching problem (MWBMP). For online MWBMP, it is known that, under some assumptions on the payoffs, a greedy algorithm has a *competitive ratio* of $\frac{1}{3}$. Our key result is to prove that for the general problem, under the same assumptions on the payoff as in MWBMP and an assumption on the number of tasks arising in each group, a repeated auction algorithm, where each group of tasks is (near) optimally allocated to the available group of robots has a guaranteed competitive ratio. We also prove that (a) without the assumptions on the payoffs, it is impossible to design an algorithm with any performance guarantee and (b) without the assumption on the task profile, the algorithms that can guarantee a feasible allocation (if one exists) have arbitrarily bad performance in the worst case. Additionally, we present simulation results depicting the average case performance of the repeated greedy auction algorithm.**

***Index Terms*— Multi-robot assignment, Task allocation, Auction algorithm, Online algorithm, Competitive analysis.**

I. Introduction

In many multi-robot applications like environmental monitoring, search and rescue, disaster response, extraterrestrial exploration, the tasks that the robots need to perform are not known beforehand but arise as the robots are executing their missions. In such scenarios, robots may be able to do more than one task during a mission depending on their capabilities and battery life. Since battery life for a robot is limited there will be an upper bound on the number of tasks that a robot can do during a mission. The problem of allocating tasks to robots when the tasks are not known beforehand but may arise in an *online* fashion is called the *online task allocation (OTA) problem* or *online assignment problem*. Depending on the characteristics of the tasks and the capability of the robots, different versions of the OTA problem can be formulated (see [1] for a classification and taxonomy of task allocation problems). In the simplest version of OTA, also known as online maximum weight bipartite matching problem (MWBMP), the tasks arrive one at a time and each robot can do at most one task in the mission. Each robot-task pair has a certain payoff and the objective is to maximize the total payoff of the multi-robot system [2], [3]. In this paper we study a generalization of the online MWBMP, where the tasks can arise dynamically in groups and each robot can do at most one task in each group, but can do more than one task in the whole mission. The abstract problem is motivated by two different kinds of scenarios arising in applications: (a) Tasks arise dynamically in groups, where each group consists of tightly-coupled tasks, i.e., tasks which robots must perform simultaneously, and thus each robot can only be assigned to one of them; (b) There exist group precedence constraints among tasks, i.e., only after the current group of tasks are all completed by robots, the subsequent group of tasks can get started, and the corresponding (payoff) information is revealed to robots. To fully explore the parallelism, each robot can be assigned to at most one task in each group to increase the efficiency.

More formally, the OTA problem studied in this paper is as follows: *We have a set of n_r robots, R, and a set of n_t tasks, T, that arrives dynamically in groups over n_s rounds. Each robot has a budget, $N_i, i = 1,\dots,n_r$, i.e., an upper bound on the maximum number of tasks that it can do. Each robot can do at most one task from each group and the execution of one group of tasks starts after the previous group has been executed. Find an assignment of the tasks to the robots such that the total payoff of the system is maximized.* Note that when $N_i = 1$ for each robot, i, and there is one task in each group, the problem is the online MWBMP. A greedy algorithm where the incoming task is assigned to the best available robot has a competitive ratio (the ratio of the payoff obtained from the greedy assignment to the payoff obtained from optimal assignment if all tasks were known beforehand) of $\frac{1}{3}$ under an assumption on the payoffs [2]. The assumption states that the difference of the payoffs of any two robots for a task is less than the sum of the payoffs of the same two robots for any other task. Furthermore, this is the best achievable bound by any online deterministic algorithm. One assumption in this work is that once a robot is assigned to a task, it cannot be reassigned, or each robot can only do one task during the mission. We consider a more general setting where a robot can do multiple tasks during its mission.

Our results for the OTA problem are a combination of positive and negative results. We first study the performance of the repeated greedy auction algorithm, where for each group of tasks, the robots are allocated to the tasks using a (distributed) auction algorithm. We prove that under the same assumptions on payoff as for the online MWBMP and an assumption on the number of tasks in each task subset, the repeated greedy auction algorithm has a competitive ratio

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

of $\frac{1}{1+\max(2,\alpha)}$. The problem data dependent parameter α is defined as the minimum of the maximum budget of the robots and the maximum number of tasks in a group. Note that the competitive ratio is independent of the number of robots or the number of tasks. Furthermore, when either the size of the task groups or the maximum budget of a robot is constant, α is constant, and hence the competitive ratio is constant. For example, when the number of tasks in each group is 2 and/or each robot can perform at most 2 tasks, the competitive ratio of the algorithm becomes $\frac{1}{3}$. This generalization of the results in [2] is the key contribution of this paper. We also prove that if there are no restrictions on the payoffs, it is impossible to design a randomized/deterministic algorithm with provable performance guarantees. If the assumption on the task profile is violated then the algorithms based on *highest budget heuristic* that can give a feasible assignment (if one exists) have arbitrarily bad worst case performance. In highest budget heuristic, when a group containing k tasks arise, they are assigned to the robots with the top k budgets (where the budget of a robot is the number of remaining tasks that it can perform). The offline version of the problem that we study here has been studied in [4] and can be solved (near) optimally in polynomial time. We present simulation results to compare the performance of our online algorithm to the optimal offline solution on randomly generated instances.

This paper is organized as follows: In Section II, we present the related work. In Section III, we give a formal definition of the online multi-robot assignment problem for groups of tasks. In Section IV, we present the repeated auction algorithm and prove its performance guarantees. Thereafter, in Section V, we present the highest budget heuristic. In Section VI, we demonstrate the performance of our algorithm with some example simulations. Finally, in Section VII, we present our conclusions and outline future avenues of research.

II. Related Work

Task allocation is important in many applications of multi-robot systems, e.g., multi-robot routing [5], multi-robot decision making [6], and other multi-robot coordination problems (see [7], [8]). 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 [1] for a taxonomy of task allocation problems). One axis of dividing the task assignment problem is as online versus offline. In offline task allocation the set of tasks are known beforehand, whereas in online problems the tasks arise dynamically. In this paper we will consider the online task allocation problem and therefore we will divide our discussion of the relevant literature here into the offline and online task allocation problems. Moreover, our objective is to design algorithms for task allocation with provable performance guarantees. Therefore, we will elaborate on algorithms that provide performance guarantees.

Offline Task Allocation: In offline task allocation, the payoff's of a robot for each task is assumed to be known beforehand. In the simplest version of the offline 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. This problem can be solved in a centralized manner using the Hungarian algorithm [9], [10]. Bertsekas [11] gave a decentralized algorithm (assuming a shared memory model of computation, i.e., each processor can access a common memory) that can solve the linear assignment problem *almost optimally*. In subsequent papers, the basic auction algorithm was extended to more general task assignment problems with different number of tasks and robots and each robot capable of doing multiple tasks [11], [12]. Recently, [13], [8] have combined the auction algorithm with consensus algorithms in order to remove the shared memory assumption and obtain a totally distributed algorithm for the basic task assignment problem. However all of this work assume that the tasks are independent of each other. For the more general case, 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, [4] generalized the auction algorithm of [11] to give an algorithm with near optimal solution. The problem studied in this paper is the online version of the problem in [4].

For multi-robot routing problems, [5] has given different auction algorithms with performance guarantees for different team objectives. When the objective is to minimize the total distance traveled by all the robots they provide a 2-approximation algorithm. For all other objectives the performance guarantees are linear in the number of robots and/or tasks. For example, when allocating m spatially distributed tasks to n robots, for minimizing the maximum distance traveled by a robot, their algorithm gives a performance guarantee of $O(n)$.

Online Task Allocation: Even the simplest version of the online task allocation problem, which is (a variation of) the online linear assignment problem is NP-hard [1]. As stated before, this is the online MWBMP where the edge weights are revealed randomly one at a time, i.e., the tasks arrive randomly and a robot already assigned to a task cannot be reassigned. Greedy algorithms for task allocation, wherein the task is assigned to the best available robot has been used in a number of multi-robot task allocation systems (e.g., MURDOCH [3], ALLIANCE [14]) and therefore, have the same competitive ratio of $\frac{1}{3}$ as [2], if the payoff's are non-negative and satisfy the Equation 6. Note that the greedy algorithm gives a solution that is exponentially worse in the number of robots, when the objective is to minimize the total payoff [2]. This is different from the offline linear assignment problem where both the maximization and minimization problems can be solved optimally in polynomial time.

There are other variations of the task allocation problem studied in the multi-robot task allocation community, as well as operation research community that have been shown to be NP-hard, and for many of them there are no algorithms with worst case approximation guarantees [1]. Therefore, a substantial amount of effort has been invested in developing and testing heuristics for dynamic task allocation [15], [16], [17]. These algorithms are based on distributed constraint optimization (DCOP). Auction-based heuristics for multi-robot task allocation in dynamic environments have also been proposed, where the robots may fail during task execution and the tasks need to be reassigned [18], [19].

III. Problem Formulation

In this section, we give the formal definition of our online multi-robot task assignment problem (denoted as "OTA").

A. Definition of the Problem OTA

Basic Multi-robot Assignment Problem (MAP): Suppose that there are n_r robots, $R = \{r_1, \dots, r_{n_r}\}$, and n_t tasks, $T = \{t_1, \dots, t_{n_t}\}$, for the robots. In *MAP*, any robot can be assigned to any task, and each robot can perform at most N_i tasks. Performing each task needs a single robot, so $n_t \le \sum_{i=1}^{n_r} N_i$. Let f_{ij} be the variable that takes a value 1 if task, t_j, is assigned to robot, r_i, and 0 otherwise. 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. The objective in *MAP* is to assign all tasks to robots to maximize the total payoff.

Task Group Constraint (TGC): The task set T is divided into n_s disjoint groups/subsets $\{T_1, \dots, T_{n_s}\}$ so that $\cup_{i=1}^{n_s} T_i = T$, and each robot can perform at most one task from each subset.

Online Multi-robot Assignment Problem with Task Group Constraint Combining the *TGC* constraint with *MAP*, the online task allocation (OTA) problem is:

Problem 1: Given n_r robots, n_s disjoint subsets of tasks that arise one at a time, assign robots to the dynamically-arising subsets of tasks (as they arise with no modification of assignments later), such that the total payoffs of robot-task assignment is maximized. Each task is performed by one robot, and each robot r_i performs at most one task from each subset and at most N_i tasks in the whole mission.

Problem 1 can be written as an Integer Linear Programming (ILP) problem:

$$\begin{aligned}
\max_{\{f_{ij}\}} \quad & \sum_{i=1}^{n_r}\sum_{j=1}^{n_t} a_{ij} f_{ij} & \\
\text{s.t.} \quad \sum_{i=1}^{n_r} f_{ij} &= 1, \ \forall j = 1, \dots, n_t & (1)\\
\sum_{t_j \in T_k} f_{ij} &\le 1, \ \forall i,k : i = 1, \dots, n_r, k = 1, \dots, n_s & (2)\\
\sum_{j=1}^{n_t} f_{ij} &\le N_i, \ \forall i = 1, \dots, n_r & (3)\\
f_{ij} &\in \{0,1\}, \ \forall i,j & (4)
\end{aligned}$$

The above objective function is the total payoff of all assignments. Constraint (1) implies that each task can be done by exactly one robot and all tasks must be performed. Constraint (2) and (3) mean that each robot can perform at most one task from each subset, and at most N_i tasks in the whole mission.

The problem OTA shares the same ILP formulation as its offline counterpart studied in [4]. However, here the payoffs a_{ij} are not revealed to robots beforehand, and instead, the payoff information related to tasks in subset T_k, $\{a_{ij} | t_j \in T_k\}$, is revealed to robots only after all preceding subsets of tasks have been performed. In other words, robots get the payoff information $\{a_{ij} | t_j \in T_k\}$ for round k ($k \in \{1, 2, \dots, n_s\}$), at the beginning of the round. Note that if $n_s = n_t$, i.e., each subset contains only one task, constraint (2) can be removed since (3) would imply (2). Thus, the online MWBMP [2], [20] (where $N_i = 1, \forall i$) and online transportation problem [21] are special cases of our problem.
Payoff Constraints: Following the online MWBMP literature [2], [20], we assume that the payoffs $\{a_{ij}\}$ are non-negative:

$$a_{ij} \ge 0, \quad \forall i, j, \tag{5}$$

and satisfy the inequality below:

$$a_{i_1 j_1} + a_{i_2 j_1} \ge |a_{i_1 j_2} - a_{i_2 j_2}|, \quad \forall i_1, i_2, j_1, j_2. \tag{6}$$

Equation (6) implies that the payoff difference of assigning any two robots to any task, is bounded by the payoff sum of assigning the same two robots to any task. This condition has an intuitive geometric interpretation. If we associate a point in a metric space with each robot and each task, and assume a_{ij} to be the Euclidean distance between robot r_i and task t_j, then the above inequality (6) can be derived from the triangle inequality in the metric space. We use this intuition later to generate payoffs that satisfy inequality (6) for our simulations in Section VI. Note that the inequality (6) does not give any bound on the ratio of minimum possible to maximum possible payoff, which can be arbitrarily high. If the payoff constraints (5) and (6) are removed, any online algorithm (either deterministic or randomized algorithms) would lead to arbitrarily bad solution in the worst case (see Appendix for details).
Constraints of Task Group Size: Depending on the size of N_i and the number of tasks in each group (or task group size), Problem 1 may not have a feasible solution, i.e., there may be tasks that remain unassigned. Let the sequence of task group sizes that arise during OTA be called a *task profile*. In this paper, we are interested in task profiles where there is a feasible assignment. An algorithm that is guaranteed to find a feasible solution if one exists is called a complete algorithm. As we will show later, for OTA, *any algorithm that is complete performs arbitrarily bad in the worst case*. We will now present some sufficient conditions on the task profile under which we can guarantee feasible task allocation. We call this constraint the *Step Constraints of Task Subset Size* (SCTSS).

Suppose that we have sorted the initial budgets of all robots in the ascending order: $N_1 \le \dots \le N_{n_r}$. The SCTSS is as follows: *The size of the k^{th} task subset, $|T_k|$, should satisfy $|T_k| \le n_r - i$ when $N_i < k \le N_{i+1}$, where $N_0 = 0$.*

The SCTSS defined above is consistent with the implicit constraints of Problem 1 that the size of each subset must

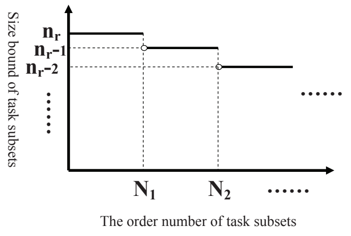

Fig. 1. Illustration of step constraints of task subset size. If the subset size at each round is under the step bounds, the step constraints are satisfied.

be not bigger than the number of robots, i.e.,

$$|T_k| \leq n_r, \forall k = 1, \dots, n_s \tag{7}$$

Furthermore, the step constraints extend the implicit constraint (7) to guarantee that when each subset of tasks T_k arises, there always exist sufficient number of different robots with non-zero remaining budget to be assigned to tasks in the subset, as proved below.

Lemma 1: Step Constraints of Task Subset Size are sufficient to guarantee that any algorithm (which assigns different robots to tasks in each subset) would lead to a feasible solution, regardless of the task payoff profile.

Proof: Denote $n_r^{(k)}$ as the number of robots with non-zero budget when task subset T_k arises. First, we observe that it is impossible to exhaust robot r_i's budget before T_{N_i+1} arises. So $\forall i$, when $k \in (N_i, N_{i+1}]$, robot $r_{i+1}, r_{i+1}, \dots, r_{n_r}$ must still be available to be assigned to one task in T_k. So $n_r^{(k)} \geq n_r - i$. From definition of SCTSS, $|T_k| \leq n_r - i$. We get that $n_r^{(k)} \geq |T_k|$, which means when each subset of tasks T_k arises, there always exist sufficient number of different robots with non-zero remaining budget to be assigned to tasks in the subset. So any algorithm, which assigns different robots to tasks in each subset, would lead to a feasible solution.■

Note that since the size of each subset is 1 in [2], [20], [21], any algorithm gives a feasible solution. In Problem 1, finding a feasible solution depends on the initial budget of robots and the size of each task subset, and it is independent of the payoff profile. The SCTSS guarantees that any algorithm would give a feasible solution. We will now show that using a greedy repeated auction heuristic to solve the OTA problem gives a solution within certain ratio of the optimal off-line solution, for any profile of payoffs and subset sizes (satisfying step constraints) in the worst case.

IV. Greedy Auction Online Algorithm

In this section, assuming that the constraints on the payoffs and task profiles hold, we present an online algorithm using *greedy auction heuristic*, and prove that the algorithm can achieve a competitive ratio of $\frac{1}{1+\max(2,\alpha)}$, where α is the minimum of the maximum budget of a robot and the maximum task group size. Subsequently, in Section V, we prove that if there exist no constraints of task subset size (except that the size of each subset is less than the number of robots), an online algorithm for OTA is complete, if and only if it uses *the highest-budget heuristic*. However, in the worst case, such online algorithm can still lead to very bad performance compared to the optimal offline solution.

The basic idea of the greedy auction heuristic is as follows: for each dynamically arising group of tasks (i.e., during each round), use auction algorithm to assign robots with non-zero *budgets* (the *budget* of robot r_i is the number of remaining tasks that can be assigned to r_i) to tasks. This guarantees that the total payoffs gained by the selected robots would be near-optimal among all possible assignments using available robots at that round [4]. Below, we first briefly review the basic idea of the auction algorithm.

The payoff information related to tasks in the kth round, T_k, is revealed to the robots at the beginning of of round k, and we want to match $n_r^{(k)}$ robots (the number of robots with non-zero remaining budget at step k, $n_r^{(1)} = n_r$) to $|T_k|$ tasks through a market auction mechanism, 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 task, the number of tasks in each subset is not bigger than the number of robots according to (1) and (2), and the different interest of robots will probably cause conflicts. This can be resolved through the auction mechanism of bidding for tasks. In round k, the robots *iteratively* bid for the tasks in the set T_k. The price for task t_j at iteration t is $p_j(t)$, which is the highest bid from robots at iteration t, and the robot assigned to the task must pay $p_j(t)$. Thus, at iteration t the net value of task t_j to robot r_i is $a_{ij} - p_j(t)$. During each iteration, every unassigned robot bids for the task with the highest net value to it, which increases the task price. The iterative bidding from robots leads to the evolution of $p_j(t)$, which can gradually resolve the interest conflicts among robots and lead to near-optimal solution of the overall assignment.

So, for each round, k, we can use the auction algorithm for $n_r^{(k)}$ robots to be assigned to $|T_k|$ tasks. Since $n_r^{(k)} \geq |T_k|$ (according to Lemma 1), we need to add $n_r^{(k)} - |T_k|$ virtual tasks with small equal payoffs to all robots. The requirement that each robot must know the current price $p_j(t)$ for all task t_j during bidding implies the existence of a centralized auctioneer or a shared memory for all robots to access. In [4], [13], [8], for a connected multi-robot network, the auction algorithm has been combined with a maximum consensus technique so that the algorithm becomes totally distributed without any centralized auctioneer. During each iteration t, each robot, r_i, in the connected network locally maintains and updates a list of current highest bids, $p_j^i(t)$, for each task, t_j, from its own neighborhood $\mathscr{N}_{r_i}$:

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

This highest bid is used as local price of tasks. Since the network is connected, the global highest bids eventually propagates to all robots so that the solution quality remains the same as that of original auction algorithm.

A single iteration of the auction algorithm for each robot r_i at round k is described in Algorithm 1. All robots run copies of Algorithm 1 sequentially. The algorithm terminates when all robots have been assigned to their tasks (i.e., $P = p_I^i(t+1)$ for each robot r_i when its turn comes).

Algorithm 1 Auction Iteration for Robot r_i with non-zero Budget at Step k

```
1: Input: $T_k$, $a_{ij}$, $p_j^\ell(t)$ for all $j,\ell : t_j \in T_k, r_l \in \mathcal{N}_{r_i}$,
   $<I,P>$ // I: index of the task assigned to $r_i$ during
   // $r_i$'s previous iteration;
   // P: the corresponding bidding price from $r_i$
2: // Update the local highest bid information:
3: for all $t_j \in T_k$ do
4:     $p_j^i(t+1) = \max_{r_\ell \in \mathcal{N}_{r_i}} p_j^\ell(t)$
5: end for
6: // Update the assignment information:
7: if $P < p_I^i(t+1)$ then
8:     // another robot has bid higher than $r_i$'s previous bid
9:     Set $I$ and $P$ to be zero
10:    // Collect information for new bids
11:    Denote $v_j(t+1) = a_{ij} - p_j^i(t+1)$ // value of $t_j$ to $r_i$
12:    Select the best candidate task $t_{j_k^*}$ from $T_k$, where $j_k^* = \arg\max_{j \in T_k} v_j(t+1)$
13:    Store the index of second best candidate from $T_k$:
       $j_k' = \arg\max_{j \in T_k, j \neq j_k^*} v_j(t+1)$
14:    // Start new bids
15:    Bid for $t_{j_k^*}$ with price:
16:    $b_{j_k^*} = p_{j_k^*}^i(t+1) + v_{j_k^*}(t+1) - v_{j_k'}(t+1) + \varepsilon$
17:    // Update assignment information and price information:
18:    Set $I = j_k^*$, $P = b_{j_k^*}$
19:    Set $p_{j_k^*}^i(t+1) = b_{j_k^*}$
20: end if
```

Algorithm 1 can be summarized as follows. First, robot r_i updates its local price list of all tasks by maximizing the price of each task in the lists of its neighbors (lines 2 to 5). Then, it updates its assignment information from its previous iteration, since other robots may bid higher price for its assigned task after its previous iteration (lines 6 to 9). If that is the case, the previous assignment of task t_I for r_i will be broken and r_i makes a new bid. During the bidding part of Algorithm 1 (lines 10 to 20), robot r_i bids for the task with the best values from the current subset T_k. This guarantees that after the iteration, all constraints in the problem are satisfied: (a) robot r_i is assigned to one task of the subset since it either switches to a new task or its previous assignment is unchanged (please note we have introduced some virtual tasks to the subset, r_i is in fact assigned to at most one task of the subset); (b) 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. The bidding price for the task is at least ε bigger than its price at the beginning of the iteration: since j_k^* is the best candidate task in T_k, j_k' is the second best in T_k,

$$b_{j_k^*} - p_{j_k^*}(t+1) = v_{j_k^*}(t+1) - v_{j_k'}(t+1) + \varepsilon \geq \varepsilon$$

. Thus, the task receiving r_i's bids must be assigned to r_i at the end of the iteration. The rule for setting the bidding value of $b_{j_k^*}$ is related to the proof of optimality of the algorithm (please refer to [11] for details).

The auction algorithm during each round, k, guarantees a near-optimal assignment for that round.[1] However, repeatedly applying the algorithm for each round of tasks does not guarantee that the whole assignment is optimal. We now present the competitive ratio of the repeated auction algorithm. Let $\alpha = \min(\max_i N_i, \max_k |T_k|)$.

Theorem 1: Under the step constraints of task subset size, the online sequential auction algorithm, alg_1, will output an assignment solution for OTA with total payoff $A \geq \frac{1}{1+\max(2,\alpha)} A^*$ in the worst case, where A^* is the solution by the optimal algorithm O for OTA after all payoff information has been revealed.

Proof: Suppose that we have relabeled the tasks so that the assignment by alg_1 is (r_i, t_i). Let's consider an assignment (r_i, t_i) at step k. Suppose that algorithm O assigned the task t_i to a different robot $r_{i'}$. Below we need prove that the payoff difference between a_{ii} and $a_{i'i}$, by assignments of alg_1 and O, are not too big.

Assume that $r_{i'}$ is different from r_i. When we consider the assignment of task t_i by algorithm alg_1, it can be divided into four cases depending on the assignment status of $r_{i'}$ in the procedure by alg_1 at the time of t_i's assignment:

(a) robot $r_{i'}$ still has non-zero budgets by alg_1 and it is not assigned to any other tasks in T_k by alg_1: in this case, we know that $a_{ii} \geq a_{i'i}$, otherwise the auction algorithm would have assigned $r_{i'}$ to t_i. If all assignments of alg_1 and O belong to this case, then $A = \sum_i a_{ii} \geq \sum_{i'} a_{ii'} = A^*$

(b) robot $r_{i'}$ still has non-zero budgets by alg_1 and it is assigned to another task $t_{i'}$ in T_k by alg_1: in this case, $a_{ii} + a_{i'i'} \geq a_{ii'} + a_{i'i}$. Since $a_{ii'} \geq 0$, we have $a_{i'i} - a_{ii} \leq a_{i'i'}$. If all assignments of alg_1 and O belong to this case, we have that $A^* = \sum_i a_{i'i} \leq \sum_i a_{ii} + \sum_{i'} a_{i'i'} = 2 * A$. So $A \geq \frac{1}{2} A^*$

(c) robot $r_{i'}$ has exhausted all its budgets by alg_1, so it must be assigned to other tasks before the subset $|T_k|$ arrives. Suppose that $r_{i'}$ was assigned to a task $t_{i'}$ in $T_{k'}$, there can be two cases:

(c.1) robot r_i is also assigned to a task t_j in $T_{k'}$; using the metric constraints,

$$\begin{aligned} a_{i'i} - a_{ii} &\leq \min(a_{i'i'} + a_{ii'}, a_{i'j} + a_{ij}) \\ &\leq \frac{1}{2}(a_{i'i'} + a_{ii'} + a_{i'j} + a_{ij}) \end{aligned}$$

According to the property of auction algorithm, $a_{ii'} + a_{i'j} \leq a_{i'i'} + a_{ij}$, so

$$\begin{aligned} a_{i'i} - a_{ii} &\leq \frac{1}{2}(a_{i'i'} + a_{ii'} + a_{i'j} + a_{ij}) \\ &\leq \frac{1}{2}(2(a_{i'i'} + a_{ij})) \\ &= a_{i'i'} + a_{ij} \end{aligned}$$

[1]For simplicity of discussion, we can assume that the assignment is optimal, since it won't change the following results.

So summing over each task t_i on the left would lead to sum over each task $t_{i'}$ and t_j on the right, corresponding to task t_i,

$$\sum_i (a_{i'i} - a_{ii}) \leq \sum_{i'} a_{i'i'} + \sum_j a_{ij}$$

When i traverses through all tasks, i' would also traverse all tasks, so $\sum_{i'} a_{i'i'} = A$.

$$A^* \leq 2A + \sum_j a_{ij}$$

, where each t_j corresponds to each task t_i. Now consider at most how many times a specific a_{ij} can repeat in $\sum_j a_{ij}$, which is bounded by how many times the robot $r_{i'}$ can be assigned to a task in $|T_{k'}|$: Since each robot i can be assigned for at most $N_i - 1$ times by alg_1 to other tasks than t_i, and this case can be bounded by the largest size of a subset $|T_{k'}| - 1$ (recall that r_i has been assigned to t_j in $T_{k'}$), so a single specific a_{ij} can repeat at most $\min(\max_i(N_i - 1), \max_k(|T_k| - 1))$ times in $\sum_j a_{ij}$. So $\sum_j a_{ij} \leq A * \min(\max_i(N_i - 1), \max_k(|T_k| - 1))$. So $A \geq \frac{1}{1+\alpha} A^*$
(c.2) robot r_i was not assigned to any task in $T_{k'}$: using metric constraints, we have that $a_{i'i} - a_{ii} \leq a_{ii'} + a_{i'i'}$. Besides, according to the property of auction algorithm, $a_{ii'} \leq a_{i'i'}$. So $a_{i'i} - a_{ii} \leq 2a_{i'i'}$. If this case is general, then $\sum_i (a_{i'i} - a_{ii}) \leq 2\sum_{i'} a_{i'i'}$, which means $A^* = \sum_i a_{i'i} \leq \sum_i a_{ii} + 2\sum_{i'} a_{i'i'} = 3 * A$. So $A \geq \frac{1}{3} A^*$.

Since $\forall t_i$, at the time of assignment of task t_i, it must belong to one case above. We get that $A \geq \min(1, \frac{1}{2}, \frac{1}{1+\alpha}, \frac{1}{3}) A^*$. So the competitive ratio of the greedy auction algorithm is $\frac{1}{1+\max(2,\alpha)}$. ■

Note that this result is consistent with the result in [2], [20], where $\max_i N_i = \max_k |T_k| = 1$. The competitive ratio is independent of the number of robots or the number of tasks. Furthermore, when either the size of the task groups or the maximum budget of a robot is constant, the competitive ratio is constant. For example, when the number of tasks in each group is 2 and/or each robot can perform at most 2 tasks, the competitive ratio of the algorithm becomes $\frac{1}{3}$. The results in this section can further be extended to a more general setting, where different task groups can arrive simultaneously in the same round, which we leave the details for a future full version.

V. Highest Budget Heuristic for OTA

In this section, we present the highest budget heuristic (HBH) and show that when the assumptions regarding the task sizes in each group is removed, any online algorithm is complete (i.e., the algorithm is guaranteed to find a feasible solution if one exists) iff it uses the HBH. Let T_k be the task set for round k. In the HBH, during each round k, the tasks are assigned to the robots with the top $|T_k|$ remaining budgets. As there can be multiple robots with the same remaining budgets, there can be different variations of HBH heuristic depending on how ties are broken. For example, if there are more than $|T_k|$ candidate robots, then we can assign the robots randomly to the task or use an auction algorithm for the assignment.

Theorem 2: Any online algorithm is complete for OTA iff it uses the highest-budget heuristics (HBH).
Proof: Due to space constraints, we provide a sketch of the proof and leave the details for a future full version. For the necessary condition, consider the step k_0 when an online algorithm A starts not to use HBH. We can construct an instance so that during each following step $k > k_0$,the size of task subset equals $n_r^{(k)}$ by HBH, i.e., the number of robots with non-zero budgets. In this instance, HBH can find a feasible solution, while A cannot since A would exhaust a robot (which A assigns a task to at step k) at certain step k_i earlier than HBH, and thus becomes infeasible at step $k_i + 1$.

For the sufficient condition, the key idea here is that HBH online algorithm would maximize the number of robots with non-zero remaining budgets, $n_r^{(k)}$ at each step k. The reason is that during each step, whenever a robot with lower budget is assigned, HBH would guarantee that the robots with higher budget must also be assigned. So when a robot r_i is exhausted by HBH at step k, modifying previous assignments of HBH cannot transfer the budgets of robots with budgets bigger than 1 at step k to robot r_i and thus cannot increase $n_r^{(k)}$.■

Theorem 3: Without constraints of task subset size, any complete online algorithm (i.e., algorithms using HBH) has arbitrarily bad performance in the worst case.
Proof: We prove this theorem by constructing a worst-case example below. Consider robots $\{r_1, r_2, \ldots r_n\}$, where $N_1 = n+1$, while $N_2 = \ldots = N_n = 1$; task subsets arrive in the order of $\{t_1\}, \{t_2\}, \ldots, \{t_{2n}\}$. Suppose $\forall t_j : j \leq n$, the payoffs $a_{1j} = 0, a_{ij} = 1 (\forall i \neq 1)$; $\forall t_j : j \geq n+1$, the payoffs $a_{1j} = 1, a_{ij} = 0 (\forall i \neq 1)$. So HBH would assign tasks t_1, t_2, t_{n+1} to r_1 and the rest of tasks to other robots, with total payoffs 1, while the optimal offline solution would assign tasks t_1, t_2, t_{n-1} to $r_2, \ldots, r_{2n}$ and the rest of tasks to r_1, leading to total payoffs $2n - 1$. So the competitive ratio is $\frac{1}{2n-1}$, which would become very bad with the number of tasks increasing. Note that the ratio is not bounded by the budget of robot r_1, since we can add any number of robots with same payoffs as r_1 to average the total budgets of r_1. ■

Theorems 2 and 3 together imply that although HBH is complete, there is no worst case performance guarantee.

VI. Simulation Results

In Section IV, we derived an expression for the worst-case competitive ratio of the repeated auction algorithm for the OTA problem. In this section, we present simulation results on random instances of a synthetic example to compare the online algorithm solution to the near-optimal off-line solution presented in [4]. The performance guarantee of our algorithm is dependent on the problem instance data parameters, namely, the maximum budget of a robot, the maximum number of tasks in a group, and the payoffs. In the auction algorithm the minimum amount by which a bid should be updated, i.e. ε, is also a parameter. Since we are interested in the performance of the algorithm as the problem data is varied, for the results presented here we assume $\varepsilon = 0.1$.

We consider $n_r = 20$ robots, where each robot performs at most $N_i = 3$ tasks during the mission from a set of $n_t = 60$ tasks. The task set T is divided into $n_s = 22$ disjoint subsets, with 3 tasks in the first 18 subsets, 2 tasks in the following 2 subsets and 1 task in the last 2 subsets. The size of task subsets are designed to satisfy the step constraints of task subset size. To generate payoffs, a_{ij}, that satisfy constraints (5) and (6), we first randomly generate some points (representing robots and tasks) in a two-dimensional 10×10 square, then use the distance between each robot and each task as the payoff of assigning the robot to the task. The points are generated as shown in Figure 2: the positions of a half robots are randomly generated in square A, while those of the other half in B, and the positions of a half tasks in earlier subsets are randomly generated in square C while those of the other half in later subsets in A. The parameter u is designed here to represent the uniformity of payoff distributions. When we change u from 0.01 to 10, the uniformity of payoff distribution increases. We generate 100 random samples for each value of u, and compute the mean and standard deviation of performance ratio of the online greedy auction algorithm over the optimal offline solution, as shown in Figure 3.

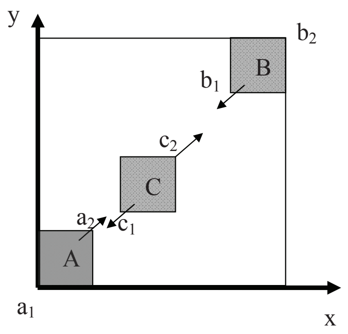

Fig. 2. The illustration of how we generate the payoffs between robots and tasks. The coordinates of end points of A: $a_1 = (0,0)$, $a_2 = (u,u)$; B: $b_1 = (10-u, 10-u)$, $b_2 = (10,10)$; C: $c_1 = (4.5-0.45u, 4.5-0.45u)$, $c_2 = (4.5+0.55u, 4.5+0.55u)$. When $u = 10$, A, B and C would become the whole 10×10 square. The arrows represent the expanding directions of the square A, B, C with parameter u increasing.

From Figure 3, we see that if the payoff distribution is uniform (e.g., $u = 10$), the performance of greedy auction algorithm would be very close to that of optimal off-line solution. The reason is that when the payoff distribution is uniform, each robot would have the same expected payoffs towards dynamically-arising tasks, so the optimal offline algorithm would do almost the same assignments as the greedy auction algorithm, since there is no need to sacrifice the payoffs of earlier assignments in hope of gaining more from later assignments. However, when u decreases, (i.e., the payoff distribution becomes more and more nonuniform), the performance ratio rapidly decreases, and approaches to $\frac{1}{3}$ when u is as small as 0.1, which is consistent with the conclusion of Theorem 1.

To see the effect of the term $\alpha = \min(\max_i N_i, \max_k |T_k|)$ in the performance bound of Theorem 1, we also test different N_i (or $\max_k |T_k|$) values (for simplicity, we set $N_i = \max_k |T_k|$ in our examples). However, the results do not change with different N_i (or $\max_k |T_k|$) as shown in Figure 3 (three examples $N_i = 2, 3, 5$ are shown in the figure, and by Theorem 1 their corresponding competitive ratio lower bounds are $\frac{1}{3}$, $\frac{1}{4}$, and $\frac{1}{6}$, respectively). There are two possible reasons: first, the third case (c.1) in the proof of Theorem 1, which leads to the term $\min(\max_i N_i, \max_k |T_k|)$, might statistically rarely exist if we randomly generate the payoffs as described above although there might exist few samples in the worst case analysis; second, the bound we proved in Theorem 1 might not be tight and can be further improved towards $\frac{1}{3}$, which we will further explore in future.

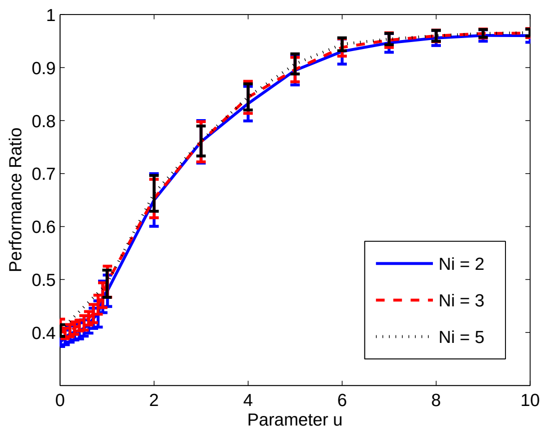

Fig. 3. The performance ratio of the solution by online greedy auction algorithm over the optimal off-line solution changes with the parameter u, representing the uniformity of payoff distribution.

VII. Summary

In this paper we introduced the online multi-robot task assignment problems (OTA), where tasks arrive in groups and have group constraints when assigning them to robots (i.e., each robot r_i can perform at most one task from each group and at most N_i tasks in the whole mission). The task group constraints distinguish our work from, and generalize previous theoretical work in online weighted bipartite matching [2], [20]. We further assume constraints of payoffs and task subset sizes, and design online algorithm based on greedy auction heuristic to achieve performance guarantee using competitive analysis. The competitive ratio is independent of the number of robots and tasks, and becomes a constant, assuming that either the size of each task group or the budget of each robot is bounded by some constant. We also show two negative results for the OTA problem (a) without the assumptions on the payoffs, it is impossible to design an online algorithm with any performance guarantee and (b) without the assumption on the task group size profile, any complete algorithm must use highest budget heuristic, which has arbitrarily bad performance in the worst case. Additionally, we present simulation results depicting the average case performance of the repeated greedy auction algorithm.

Future Work: One natural direction would be to improve the bound achieved in Theorem 1, either proving it is tight or reducing the bound towards $\frac{1}{3}$. Although worst-case analysis is crucial in some adversarial environment or situations with low tolerance of bad instances, one interesting direction would be to do average performance analysis of our online algorithms, and study which online algorithms would perform better in specific payoff (or task group size) profile.

ACKNOWLEDGMENTS

This work was partially supported by AFOSR MURI grant FA95500810356 and by ONR grant N000140910680.

APPENDIX

A. OTA Problem without Payoff Constraints

Here we remove the payoff constraints (5, 6) in OTA and show negative results of online algorithm design in this case.

Theorem 4: Any deterministic online algorithm would lead to arbitrarily bad performance for OTA without payoff constraints (5,6) in the worst case.

Proof: Assume that (a) the number of task in T_{n_s} is smaller than n_r and (b) the number of tasks equal to the total budgets of all robots. Before the last step n_s (where the last subset of tasks T_{n_s} will be assigned to robots), any deterministic algorithm would have exhausted the budgets of at least one robot r_{i^*} (since only $|T_{n_s}|$ ($|T_{n_s}| < n_r$) different robots will be assigned to tasks in T_{n_s}). So we can construct an example that there exists a task $t_{j^*} \in T_{n_s}$ such that $a_{ij^*} = -\infty \; \forall i \neq i^*$ and $a_{i^*j^*} = a_0$ where a_0 is a constant. Then in this case, any deterministic algorithm would output an assignment with total payoffs $-\infty$, while the optimal assignment would assign robot i^* to j^*, and thus have at least finite total payoffs. So we conclude that in the worst case any deterministic algorithm would have arbitrarily bad performance compared to the optimal solution. ■

From the proof above, we can see that the bad solution of deterministic algorithm is unavoidable due to the following two aspects: one is the online property that payoff information is revealed dynamically so that it is impossible to prevent the worst case mentioned above beforehand; the other is that the constraint (3) prevents robot r_{i^*} to perform task t_{j^*} even if it would incur $-\infty$ payoff.

Theorem 5: Any randomized online algorithm would lead to arbitrarily bad performance for OTA without payoff constraints (5,6) in the worst case.

Proof: The basic idea of the proof is similar to that of Theorem 4. Before the last step n_s, any randomized algorithm would have non-zero probability to assign at least one robot r_{i^*} to tasks in T_{n_s} (otherwise all tasks in T_{n_s} would not be assigned to any robot). So we can construct an example that $a_{i^*j} = -\infty \; \forall j \in T_{n_s}$. Then in this case, any randomized algorithm would output an assignment with total expected payoffs $-\infty$, while the optimal assignment would have zero-probability of assigning robot r_{i^*} to any task in T_{n_s}, and thus have at least finite total payoffs. So we conclude that in the worst case any randomized algorithm would have arbitrarily bad performance compared to the optimal solution. ■

REFERENCES

[1] 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.

[2] B. Kalyanasundaram and K. Pruhs, “Online weighted matching,” *J. Algorithms*, vol. 14, pp. 478–488, May 1993.

[3] 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.

[4] L. Luo, N. Chakraborty, and K. Sycara, “Multi-robot assignment algorithms for tasks with set precedence constraints,” in *Proceedings of IEEE International Conference on Robotics and Automation, 2011*, May 2011.

[5] 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.

[6] C. Bererton, G. Gordon, S. Thrun, and P. Khosla, “Auction mechanism design for multi-robot coordination,” in *Proc. Advances in Neural Information Processing Systems Conf.*, 2003, p. 879C886.

[7] 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.

[8] 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.

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

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

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

[12] ——, “The auction algorithm for assignment and other network flow problems: A tutorial,” *Interfaces*, vol. 20, no. 4, pp. 133–149, 1990.

[13] 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.

[14] 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.

[15] R. Nair, T. Ito, M. Tambe, and S. Marsella, “Task allocation in the robocup rescue simulation domain: A short note,” in *RoboCup 2001: Robot Soccer World Cup V*. London, UK: Springer-Verlag, 2002, pp. 751–754.

[16] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe, “Allocating tasks in extreme teams,” in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS ’05, 2005.

[17] S. Okamoto, P. Scerri, and K. Sycara, “Allocating spatially distributed tasks in large, dynamic robot teams,” in *Submitted to International Conference on Intelligent Agent Technology*, 2011.

[18] M. Nanjanath and M. Gini, “Repeated auctions for robust task execution by a robot team,” *Robotics and Autonomous Systems*, vol. 58, pp. 900–909, 2010.

[19] M. Bernardine Dias, M. Zinck, R. Zlot, and A. Stentz, “Robust multirobot coordination in dynamic environments,” in *Proceedings of 2004 IEEE International Conference on Robotics and Automation*, vol. 4, 2004, pp. 3435 – 3442.

[20] S. Khuller, S. G. Mitchell, and V. V. Vazirani, “On-line algorithms for weighted bipartite matching and stable marriages,” *Theoretical Computer Science*, vol. 127, no. 2, pp. 255–267, 1994.

[21] B. Kalyanasundaram and K. R. Pruhs, “The online transportation problem,” *SIAM J. Discret. Math.*, vol. 13, pp. 370–383, May 2000.