Auton Agent Multi-Agent Syst (2013) 26:389–419
DOI 10.1007/s10458-012-9196-7

Considering inter-task resource constraints in task allocation

Yu Zhang · Lynne E. Parker

Published online: 31 March 2012

Abstract This paper focuses on task allocation with single-task robots, multi-robot tasks and instantaneous assignment, which has been shown to be strongly NP-hard. Although this problem has been studied extensively, few efficient approximation algorithms have been provided due to its inherent complexity. In this paper, we first provide discussions and analyses for two natural greedy heuristics for solving this problem. Then, a new greedy heuristic is introduced, which considers inter-task resource constraints to approximate the influence between different assignments in task allocation. Instead of only looking at the utility of the assignment, our approach computes the expected loss of utility (due to the assigned robots and task) as an offset and uses the offset utility for making the greedy choice. A formal analysis is provided for the new heuristic, which reveals that the solution quality is bounded by two different factors. A new algorithm is then provided to approximate the new heuristic for performance improvement. Finally, for more complicated applications, we extend this problem to include general task dependencies and provide a result on the hardness of approximating this new formulation. Comparison results with the two natural heuristics in simulation are provided for both formulations, which show that the new approach achieves improved performance.

Keywords Coalition formation · Task allocation · Multi-robot systems

1 Introduction

Gerkey and Mataric [8] have categorized task allocation in multi-robot systems based on the robots (single-task or multi-task), the tasks (single-robot or multi-robot) and the assignment

Y. Zhang (✉) · L. E. Parker
Distributed Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, USA
e-mail: yzhang51@eecs.utk.edu

L. E. Parker
e-mail: parker@eecs.utk.edu

(instantaneous or time-extended). In this paper, we are focusing on the problem with single-task robots, multi-robot tasks and instantaneous assignment (ST-MR-IA). The ST-MR-IA problem requires assigning a set of tasks to a set of robots in which robots need to form coalitions for the tasks (i.e., individual robots may not have all the capabilities for a task), with the constraint that each robot and task can be assigned to no more than one coalition in the chosen assignments (i.e., coalition–task pairs). One effect of this constraint is that the capabilities on the robots are not sharable between different chosen assignments. The goal is to maximize the sum of the utilities of the chosen assignments.

The ST-MR-IA problem is closely related to the coalition formation problem in multi-agent systems. In the coalition formation problem, a set of agents replaces the set of robots and there is a function that maps a coalition of agents to a real nonnegative utility value [14]. The goal is to find a partition of this set of agents (i.e., referred to as a coalition structure in [14]) to maximize the sum of the utilities of the coalitions in the partition. Compared to the coalition formation problem, the ST-MR-IA problem is slightly different in that it also requires a notion of task and incorporates an extra constraint on the tasks. Furthermore, it is not necessary to assign every robot to some task, since there may not be suitable or sufficient tasks to be assigned.[1] Moreover, the utilities of assignments in the ST-MR-IA problem are not only dependent on the coalitions, but also on the tasks. As a result, different assignments with the same coalition can have different utilities. Due to these differences, most algorithms from the agent-based coalition formation problem cannot be directly applied to the ST-MR-IA problem.

The ST-MR-IA problem can be easily shown to be NP-hard via a reduction from the coalition formation problem,[2] which is known to be NP-hard [14]. In [8], the ST-MR-IA problem is further shown to be strongly NP-hard [7] by a similar reduction from the set partitioning problem. As a result, fully polynomial approximation algorithms for ST-MR-IA are unlikely to exist (unless $P \equiv NP$). Due to this complexity, few approximation algorithms with good solution guarantees have been provided. In this paper, we present a new heuristic that considers inter-task resource constraints in task allocation. The proposed heuristic takes into account the influence between different assignments while still maintaining polynomial running time. A formal analysis is provided for this new heuristic, which reveals that the solution quality is bounded by two different factors. Algorithms implementing this heuristic are easy to implement and simulation results show that they indeed improve the performance.

Although scheduling is not addressed in ST-MR-IA,[3] for more complicated situations involving task dependencies, the formulation of ST-MR-IA is insufficient. For example, in a disaster response scenario [10], in order for truck agents to address fires in buildings, bulldozer robots must be assigned along with the truck agents to clear city roads leading to these buildings that are blocked by impassable debris. The task to clear city roads makes the task of addressing fires possible. On the other hand, when there are alternative blocked roads that lead to the same buildings, the bulldozer robots only need to clear one of them. In this situation, the task to clear one road makes the other alternatives unnecessary (so that more bulldozer robots remain available for other tasks). It is clear that disregarding these task dependencies can significantly reduce the efficiency of the overall system. We thus extend the formulation of the ST-MR-IA problem to incorporate general task dependencies and provide

[1] In ST-MR-IA, it is also not necessary for every task to be assigned to some robot(s), since there may not be suitable or sufficient robots to assign.

[2] The reduction creates a special task for each coalition and assigns the utility value of the assignment according to the utility function, while assigning the utility values of assignments with other coalitions for the task as zeros.

[3] Scheduling is typically considered in time-extended assignment (TA).

an analysis of the complexity for the extended formulation. An algorithm that utilizes the discussed methods for ST-MR-IA is provided to address this extended formulation of the problem.

The remainder of this paper is organized as follows. After presenting a general formulation of the ST-MR-IA problem in Sect. 2, we discuss related work for addressing the coalition formation and the ST-MR-IA problems in Sect. 3. In Sect. 4, we provide a formal analysis of two natural greedy heuristics for ST-MR-IA. A new greedy heuristic and the algorithms for implementing it are discussed in Sect. 5. The extended formulation of the ST-MR-IA problem (referred to as ST-MR-IA-TD) and the result on the hardness of approximating it are presented in Sect. 6. An algorithm that utilizes the discussed methods for ST-MR-IA to address the ST-MR-IA-TD problem is provided. Simulation results for both formulations of the problem are presented in Sects. 7 and 8. Finally, we make conclusions and discuss future work in Sect. 9.

2 Problem formulation

We first provide a general formulation of the ST-MR-IA problem. This problem is often constructed similarly [17,18,20] as follows:

Given:

- a set of robots, $R = \{r_1, r_2, \ldots\}$. Each robot r_i is associated with a vector $\boldsymbol{B}_i$ of H real non-negative capabilities, in which H is assumed to be a constant.
- a set of coalitions, $C = \{c_1, c_2, \ldots\}$. Each coalition c_j satisfies $c_j \subseteq R$.
- a set of tasks to be assigned, $T = \{t_1, t_2, \ldots\}$. Each task t_l requires a vector $\boldsymbol{P}_l$ of H real non-negative capabilities.
- a vector $\boldsymbol{W}$ of real non-negative costs for capabilities: the use of the capability indexed by h incurs $\boldsymbol{W}[h]$ cost per unit.
- a vector $\boldsymbol{V}$ of real positive rewards for tasks: accomplishing task t_l receives $\boldsymbol{V}[l]$ reward.
- a function $Cost : C \times T \rightarrow \Re^0$ that computes real non-negative communication and coordination costs for assignments based on the coalition–task pair.
- a utility function U for assignments, defined as:

$$U(m_{jl}) = \begin{cases} \boldsymbol{V}[l] - \sum_h \boldsymbol{P}_l[h]\boldsymbol{W}[h] - Cost(c_j, t_l) & \text{if } \forall h : \sum_{r_i \in c_j} \boldsymbol{B}_i[h] \geq \boldsymbol{P}_l[h], \\ 0 & \text{otherwise.} \end{cases}$$

in which m_{jl} denotes the assignment of $c_j \rightarrow t_l$. Note that although mathematically, we can combine $\boldsymbol{V}[l]$ and $\sum_h \boldsymbol{P}_l[h]\boldsymbol{W}[h]$ for each task t_l as a single measure, they are often independent in robotic applications, and hence are specifically modeled for more generality.[4]

Then the problem is to maximize:

$$\sum_j \sum_l U(m_{jl})\beta_{jl} \tag{1}$$

subject to the constraints:

[4] Although not investigated in this paper, the costs of the capabilities may be dependent on the robots. For example, if the cost is related to the time spent to perform a computation, a robot with a faster processor should incur a lower cost.

$$\sum_j \sum_l \alpha_{ij}\beta_{jl} \leq 1 \quad \forall r_i \in R$$
$$\sum_j \beta_{jl} \leq 1 \quad \forall t_l \in T \tag{2}$$

in which β_{jl} is 1 if m_{jl} is in the chosen assignments or 0 otherwise, and α_{ij} is 1 if $r_i \in c_j$ or 0 otherwise. Note that the first constraint also implies that a coalition can be assigned to no more than one task in the chosen assignments.

Any assignment m_{jl} that satisfies $\forall h : \sum_{r_i \in c_j} \boldsymbol{B}_i[h] \geq \boldsymbol{P}_l[h]$ is referred to as a *feasible* assignment. In this paper, we assume that the utility function U always returns positive values for feasible assignments (to distinguish from infeasible assignments). Note that we can simply ignore feasible assignments for which the overall costs are no less than the rewards of the tasks, and for which $U(m_{jl})$ is non-positive, since they would not increase the solution quality. Henceforth, when we refer to assignments, we are always referring to feasible assignments. For example, when we state that no assignments exist, we really mean that no feasible assignments exist. Another note is that while $|C|$ can be exponential in the number robots (i.e., $2^{|R|} - 1$), which also leads to an exponential space complexity for $Cost$, reasonable assumptions are often utilized (e.g., [18]) to restrict $|C|$.

3 Related work

The coalition formation problem has been studied extensively as *characteristic function games* in multi-agent systems (e.g., [1,13,14]), which concentrate on generating optimal coalition structures. Sandholm et al. [14] show that for any algorithms to obtain solution guarantees, the search process is required to visit an exponential number of coalition structures in the number of agents. Sandholm et al. [15] also show that it is difficult to approximate the problem using techniques from combinatorial auctions. Nevertheless, researchers have proposed efficient algorithms for this problem. In [1], a novel distributed algorithm is presented that returns a solution in polynomial time, given an underlying hierarchical organization. Reinforcement learning techniques are utilized to increase the solution quality as the agents gain more experience. In [13], an efficient anytime algorithm is provided that uses a novel representation of the search space to partition the solution space and remove unpromising sub-spaces. The branch-and-bound technique is then applied to reduce the search of the remaining sub-spaces. A similar problem is the set partitioning problem, for which many algorithms have been provided (e.g., [2,9]). However, these discussed approaches cannot be utilized to address the ST-MR-IA problem due to the fact that the notion of task is absent.

The problem of coalition formation for task allocation has been studied in [4,11,17–19]. Lau and Zhang [11] have introduced a taxonomy for this problem based on three factors: demands, resources and profit objectives. They have investigated five distinct classes of the problem and have provided analyses and algorithms for each class. In their formulation, coalitions are allowed to overlap so that the same robots can potentially be assigned to multiple tasks. This approach assumes that the capabilities on the robots are sharable between different coalitions, which does not apply to multi-robot systems [21]. Note that since task locations are often geographically distant, physical robots cannot execute different tasks simultaneously. As a result, these algorithms are not suitable for addressing the ST-MR-IA problem. Dang and Jennings [4] studied the coalition formation problem in a task-based setting. They have provided an anytime algorithm that has bounded solution quality with a minimal search. Their formulation of the problem is in fact more general than ST-MR-IA, since multiple

tasks are allowed to be assigned to any coalitions. However, they did not study the influence of the size of the coalitions on the solution quality.

Meanwhile, the formulations of the problem studied in [17–19] match more with that of the ST-MR-IA problem. In [19], a fully distributed algorithm is presented, in which a maximal clique approach is applied for ensuring the high degree of communication connectivity of the candidate coalitions, thus providing more robustness. Task allocation is then achieved by selecting from these candidate coalitions based on utilities. However, the algorithm does not provide any solution guarantees. Shehory and Kraus [18] have adapted a greedy heuristic [3] from the set covering problem to address the task allocation problem via coalition formation in both multi-agent and multi-robot systems. They have studied two cases of the problem with non-overlapping and overlapping coalitions, in which the non-overlapping case is almost identical to the ST-MR-IA problem, except that a cost measure is used instead of a utility measure. A non-super-additive environment is assumed so that they can bound the size of the coalitions. With this assumption, they have shown that the greedy algorithm produces a solution that is within a logarithmic factor of the optimal.

However, in the most recent work of [17], Service and Adams point out that changing the optimization problem from a cost (as in [18]) to a utility measure has a great impact on the performance of the algorithm. They have studied two models of the problem in which the resource model is exactly the ST-MR-IA problem. It is proven in [17] that it is NP-hard to approximate the solution within $O(|T|^{1-\epsilon})$ without restricting the size of the coalitions. In addition, it is NP-hard to obtain an approximation ratio that is asymptotically no worse than $k/\log(k)$ when the maximum size of the coalitions is restricted to k, using the results reported in [23]. Service and Adams have also provided a greedy heuristic and reported an approximation ratio of $\theta = k + 1$ in the worst case. This same heuristic is presented in the next section as one of the natural heuristics (i.e., *MaxUtility*). We also analyze another natural heuristic and provide a new heuristic in this paper.

In the research of multi-robot systems, the task allocation problem has also been studied extensively [5,6,12,16,20,22] (some of these are not necessarily restricted to the ST-MR-IA problem [5,16,22]). Some approaches are designed to achieve specific objectives or tasks [6,16,22]. In [6], a backoff adaptive scheme is employed to enable fault-tolerant task allocation with uncertain task specifications. In [16], a framework for a cooperative exploration task in dynamic environments is proposed. During execution, costs are re-evaluted in the current situation based on a cost function and robots can dynamically change targets to minimize the total costs. In [22], a way to generalize task descriptions as task trees is provided, which is implemented in a distributed solution for allocating complex tasks (i.e., involving task scheduling and decomposition) using a market-based approach. For approaches that provide techniques to address the general optimization problem, anytime algorithms are applied in [5,12] and heuristics are utilized to return potentially good solutions first. In [20], Vig and Adams adapt the approach in [18] to multi-robot systems. To address the location constraints of capabilities (e.g., in a tracking task, a camera must be mounted on a robot that is also mobile), they have designed a process to check the satisfaction of these constraints and remove assignments that violate them. The same process can be applied as a pre-processing step in our approach. As an alternative, Vig and Adams have introduced the service model (also studied in [17]) to avoid this process. However, such an approach requires the services to be predefined for various tasks, which can potentially be dependent on the capabilities of the robots that are unknown.

4 Natural greedy heuristics

In this section, we present two natural greedy heuristics for addressing the ST-MR-IA problem and provide an analysis of their performances. Before continuing, we formally define *worst case ratio* (similar to the definition of *approximation factor* in [17] or *ratio bound* in [18]) used to describe the quality of approximations. In the definition, $\boldsymbol{f}$ is used to denote any computable function.

Definition 1 Given a maximization problem with solutions having positive values, an approximation algorithm has a worst case ratio $\theta = \boldsymbol{f}(I)$ ($\theta \geq 1$), if it satisfies $S^*(I) \leq \theta \cdot S(I)$ for any problem instance of I, in which $S^*(I)$ is the value of the optimal solution and $S(I)$ is the value of the solution produced by the algorithm. When $\boldsymbol{f}$ is a polynomial time computable function, the worst case ratio is also referred to as a poly-time worst case ratio.

4.1 AverageUtility

The *AverageUtility* heuristic at each step chooses the assignment that maximizes the average utility per robot, until no more assignments that satisfy the constraints in Eq. 2 exist. More formally, at step λ, denote the previously chosen set of assignments as $G_{\lambda-1}$. *AverageUtility* chooses the assignment m_{pq} that satisfies the problem constraints (given that $\forall m_{jl} \in G_{\lambda-1} : \beta_{jl} = 1$) while maximizing $\frac{U(m_{pq})}{|c_p|}$. In the following theorem, we establish the worst case ratios of this heuristic.

Theorem 1 *Applying AverageUtility to the ST-MR-IA problem yields a worst case ratio $\theta = |R|$ without restricting the size of the coalitions. Furthermore, restricting the maximum size of the coalitions to be k gives a worst case ratio $\theta = 2k$.*

Proof At the beginning of any greedy step λ, denote the remaining set of robots as R_λ, the remaining set of tasks as T_λ, and the assignment to be chosen by *AverageUtility* as $m^\lambda = (c^\lambda \rightarrow t^\lambda)$. According to the greedy criterion, m^λ has the maximum utility per robot in the remaining problem of (R_λ, T_λ). As a result, the optimal solution for (R_λ, T_λ) yields an overall utility of no more than $|R_\lambda| \frac{U(m^\lambda)}{|c^\lambda|}$. This upper bound is reached when all assignments in the optimal solution for solving (R_λ, T_λ) have a utility per robot of no less than[5] $\frac{U(m^\lambda)}{|c^\lambda|}$ and every robot in R_λ is assigned to a task. Hence, the worst case ratio for solving (R_λ, T_λ) is $\frac{|R_\lambda|}{|c^\lambda|}$. As this is true for every step, it holds true in particular for $(R_1, C_1) = (R, T)$. Consequently, the worst case ratio for *AverageUtility* can be no worse than $|R|$, given that $|c^1| \geq 1$.

When the maximum size of the coalitions is restricted to be k, we apply an induction process on the sizes of the robot and task sets. Suppose that the worst case ratio $2k$ holds for solving (R', T') in which $R' \subseteq R$, $T' \subseteq T$ and the equalities do not hold simultaneously. For solving (R, T), denote the first assignment made by *AverageUtility* as m^1, which has the maximum utility per robot. Denote the set of overlapping assignments[6] with m^1 in the optimal solution for solving (R, T) as M^* and the set of tasks in M^* as T^*. As each robot can be assigned to at most one task, we have:

$$|M^*| \leq |c^1| \tag{3}$$

[5] They cannot have more due to the greedy criterion.

[6] For two assignments m_{jl} and m_{pq}, we define that m_{jl} overlaps with m_{pq} (or vice versa) if $c_j \cap c_p \neq \emptyset$.

We use R^* to denote all robots in M^*, and R^+ to denote all robots in M^* or c^1. (Note that a robot in c^1 may not be in M^*, since the robot may not be used in the optimal solution.) Given the monotonicity of the optimal solution,[7] we have:

$$S^*(R - R^+, T - T^*) \leq S^*(R - c^1, T) \tag{4}$$

recall that $S^*(I)$ represents the optimal solution for I.

From the assumption of the induction, we have:

$$S^*(R - c^1, T - t^1) \leq 2k \cdot S^{AU}(R - c^1, T - t^1) \tag{5}$$

in which we use $S^{AU}(I)$ to denote the solution returned by *AverageUtility* for I. Also note that $(R - R^*, T - T^*)$ is a subproblem for (R, T) (so is (R^*, T^*)), in that if the set of assignments using $R - R^*$ and involving $T - T^*$ in the optimal solution for (R, T) yields a lesser or equal overall utility, it can be directly substituted by the set of assignments in the optimal solution for $(R - R^*, T - T^*)$ to create a better or equivalent solution.[8] Since robots in $R^+ - R^*$ are not present in the optimal solution for (R, T), we must have that $S^*(R - R^+, T - T^*) = S^*(R - R^*, T - T^*)$. Furthermore, for solving (R^*, T^*), the optimal solution obtains a utility no more than $k \cdot |M^*| \cdot \frac{U(m^1)}{|c^1|}$, which happens only when every task in T^* is assigned with utility per robot no less than $\frac{U(m^1)}{|c^1|}$ and is assigned to a coalition with exactly k robots. Hence, we have:

$$S^*(R, T) \leq k \cdot |M^*| \cdot \frac{U(m^1)}{|c^1|} + S^*(R - R^+, T - T^*) \tag{6}$$

From the above equations, we conclude:

$$\begin{aligned} S^*(R, T) &\leq k \cdot U(m^1) + S^*(R - c^1, T) \\ &\leq k \cdot U(m^1) + k \cdot \frac{U(m^1)}{|c^1|} + S^*(R - c^1, T - t^1) \\ &\leq 2k \cdot U(m^1) + 2k \cdot S^{AU}(R - c^1, T - t^1) \\ &\leq 2k \cdot S^{AU}(R, T) \end{aligned} \tag{7}$$

Finally, as the induction assumption holds trivially when $|R'| \leq 1$ and $|T'| \leq 1$, the conclusion holds.

Note that when k is relatively close to $|R|$, the worst case ratio for the restricted case is in fact better than $2k$. This is due to the fact that the inequalities in the proof can be further tightened in these situations. For example, when $k = |R|$ (equivalent to the unrestricted case), the worst case ratio for the restricted case is $|R|$ instead of $2|R|$. Similar effects can also be discerned in the analysis for the following heuristic.

4.2 MaxUtility

The *MaxUtility* heuristic at each step chooses the assignment with the maximum utility, until no more assignments that satisfy the constraints in Eq. 2 exist. More formally, at step λ, denote the previously chosen set of assignments as $G_{\lambda-1}$. *MaxUtility* chooses the assignment m_{pq}

[7] Given (R_1, T_1) and (R_2, T_2), if $R_1 \subseteq R_2$ and $T_1 \subseteq T_2$, we have that the overall utility of the optimal solution for (R_2, T_2) is no less than for (R_1, T_1). In fact, choosing the same set of assignments for (R_2, T_2) as in the optimal solution for (R_1, T_1) would yield the same overall utility for (R_1, T_1) and (R_2, T_2).

[8] This substitution does not influence the assignments involving T^* in the optimal solution for (R, T), since all robots involving T^* are in R^*.

that satisfies the problem constraints (given that $\forall m_{jl} \in G_{\lambda-1} : \beta_{jl} = 1$) while maximizing $U(m_{pq})$.

Theorem 2 *Applying MaxUtility to the ST-MR-IA problem yields a worst case ratio $\theta = |R|$ without restricting the size of the coalitions. Furthermore, restricting the maximum size of the coalitions to be k gives a worst case ratio of $\theta = k + 1$* [17].

Proof Service and Adams [17] have proven these worst case ratios for *MaxUtility*. It is also not difficult to conclude the same using a similar induction process as shown in the previous proof.

First of all, it is important to note that algorithms for implementing both heuristics are exponential in the number of robots (i.e., $|R|$) when the maximum size of the coalitions is not restricted and are polynomial in the order of $O(|R|^k)$ when it is restricted to k. Furthermore, it may appear at first that *AverageUtility* should yield a better worst case ratio than *MaxUtility* in the restricted case, although the theoretical results turn out to be quite to the contrary. Another note is that the worst cases in the above proofs can actually occur, such that all these proven worst case ratios are in fact tight bounds. Although it is shown that approximation algorithms with a worst case ratio asymptotically no worse than $k/\log(k)$ (which is already close to the worst case ratios of the two natural heuristics) are unlikely to exist unless $P \equiv NP$, it does not imply that algorithms with better average performance, or with better worst case ratios for certain problem instances, cannot be found.

5 The new greedy heuristic

To create a new heuristic with better average performance, we draw inspiration from the two natural heuristics. Although *MaxUtility* has a better worst case ratio than *AverageUtility* in the restricted case, as we show in the result sections, the two heuristics actually perform similarly empirically. Our explanation for this phenomenon can be understood in the proofs of their worst case ratios. To achieve the worst case ratio for *AverageUtility*, the problem instance has to be more constrained than for *MaxUtility*. In other words, it is less likely for a worst case scenario to occur for *AverageUtility* than for *MaxUtility*. Keeping this in mind, we propose a new heuristic that considers inter-task resource constraints to address the ST-MR-IA problem. Instead of making greedy choices based solely on the assignment, the new greedy heuristic also considers the influence between different assignments for task allocation.

5.1 A motivating example

As a motivating example, consider the case when we have four tasks $T = \{t_1, t_2, t_3, t_4\}$ with capability requirements $\boldsymbol{P}_1 = (1, 1, 1, 0, 0)$, $\boldsymbol{P}_2 = (1, 0, 0, 1, 1)$, $\boldsymbol{P}_3 = (0, 1, 0, 1, 1)$, $\boldsymbol{P}_4 = (0, 0, 1, 1, 1)$. Suppose that each robot has one and only one capability with a non-zero value (i.e., 1). Furthermore, suppose that we have sufficient robots for the last two capabilities and we have only one robot capable for each of the first three. Let the costs for the capabilities be the same and let $Cost$ return zeros for all assignments. In this scenario, when t_1 has a slightly higher reward than any other task, both *AverageUtility* and *MaxUtility* produce a solution in which only t_1 is assigned, hence giving a solution with poor quality. A better solution is to assign each of the three robots to tasks t_2, t_3 and t_4, respectively, which collectively yield a greater utility.

5.2 Inter-task resource constraints

From the previous example, we can see that one problem with the natural heuristics is that a task with a slightly higher reward may be assigned to robots that are essential for other tasks, which in turn sabotages the assignments of these other tasks. In light of this, we give the following definition.

Definition 2 For any two assignments m_{jl} and m_{pq}, m_{jl} conflicts with m_{pq} (or vice versa) if $c_j \cap c_p \neq \emptyset$ or $t_l \equiv t_q$. Note that based on this definition, an assignment always conflicts with itself.

This definition captures the influence of making an assignment on other assignments. In the following discussions, we refer to this influence as *inter-task resource constraints*, which are introduced by the constraints in Eq. 2. Note that not only robots, but also tasks, are considered as *resources* in this definition, since once a task is assigned, it cannot be re-assigned. As with Shehory [18], we assume non-super-additive environments so that we can restrict the maximum size of the coalitions to be k. For each assignment, we compute a measure that reflects the potential loss of utility due to conflicts with other assignments. This measure is then used to offset the utility of the assignment in consideration to produce the measure used at every greedy step.

5.3 ResourceCentric

At the beginning of any greedy step λ, R_λ and T_λ are used to represent the remaining sets of robots and tasks, respectively. The new heuristic, called *ResourceCentric*, chooses the assignment m_{xy} in (R_λ, T_λ)[9] to maximize ρ_{xy} (defined as follows), until no more assignments in (R_λ, T_λ) exist:

$$\rho_{xy} = U(m_{xy}) - \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \tag{8}$$

in which $M_{jl}(\lambda)$ represents the set of assignments conflicting with m_{jl} in (R_λ, T_λ) (note that $m_{jl} \in M_{jl}(\lambda)$, given that $m_{jl} \in (R_\lambda, T_\lambda)$). Working on (R_λ, T_λ) instead of (R, T) ensures that new assignments do not conflict with ones that are previously chosen.

First of all, at greedy step λ, since previous assignments are already made, we can only optimize on (R_λ, T_λ) by assuming it as a subproblem for (R, T). For any assignment m_{pq} in (R_λ, T_λ), it is not difficult to conclude that at least one assignment would be chosen in $M_{pq}(\lambda)$ in $S^*(R_\lambda, T_\lambda)$.[10] This is because if no assignment in $M_{pq}(\lambda)$ is chosen in $S^*(R_\lambda, T_\lambda)$, we must have that m_{pq} does not conflict with any assignments in $S^*(R_\lambda, T_\lambda)$, since all conflicting assignments are in $M_{pq}(\lambda)$. As a result, m_{pq} can be chosen to increase the overall utility for (R_λ, T_λ), which leads to a contradiction as $m_{pq} \in M_{pq}(\lambda)$. Without prior knowledge of the optimal solution and hence assuming that all assignments are equally likely to be chosen, every assignment in $M_{pq}(\lambda)$ at least has a probability of $\frac{1}{|M_{pq}(\lambda)|}$ to be in $S^*(R_\lambda, T_\lambda)$. This holds in particular for $m_{pq} \in M_{pq}(\lambda)$. As choosing m_{xy} would exclude all assignments in $M_{xy}(\lambda)$ from further consideration, hence comes the subtraction term in Eq. 8.

Lemma 1 *For two steps λ and γ ($\lambda, \gamma \in \mathbb{Z}^+$) in the greedy process, given any assignment m_{jl} and that $\lambda \leq \gamma$, we have $|M_{jl}(\lambda)| \geq |M_{jl}(\gamma)|$.*

[9] An assignment m_{xy} is in (R_λ, T_λ) if $c_x \subseteq R_\lambda$ and $t_y \in T_\lambda$, denoted by $m_{xy} \in (R_\lambda, T_\lambda)$.

[10] $S^*(I)$ is overloaded henceforth to also denote the set of chosen assignments in the optimal solution for I when there is no ambiguity.

Proof First of all, given that $\lambda \leq \gamma$, we must have $R_\gamma \subseteq R_\lambda$ and $T_\gamma \subseteq T_\lambda$. As a result, for any assignment, the number of conflicting assignments with it in step γ cannot be greater than that in step λ. This is due to the fact that any assignment in (R_γ, T_γ) would also be in (R_λ, T_λ). Hence, we have $|M_{jl}(\lambda)| \geq |M_{jl}(\gamma)|$.

Lemma 1 establishes the relationships between the scalars (i.e., $|M_{jl}(\lambda)|$) in Eq. 8 in different steps of the greedy process. The following lemma provides a way to connect the solution returned by the greedy process to the optimal solution.

Lemma 2 *Let G represent the set of assignments returned by ResourceCentric, m^λ represent the assignment chosen at step λ, and $\mathscr{M}$ represent all assignments. We have:*

$$\sum_{m^\lambda \in G} \sum_{m_{jl} \in M^\lambda(\lambda)} \boldsymbol{f}(m_{jl}) = \sum_{m_{jl} \in \mathscr{M}} \boldsymbol{f}(m_{jl}) \tag{9}$$

in which $M^\lambda(\lambda)$ represents the set of assignments conflicting with m^λ in (R_λ, T_λ), and $\boldsymbol{f}(m_{jl})$ represents any function that is only dependent on m_{jl} given the problem instance.

Proof At any step λ, first note that since $M^\lambda(\lambda)$ includes all assignments that conflict with m^λ in the remaining problem of (R_λ, T_λ), assignments in $M^\lambda(\lambda)$ are removed from consideration after m^λ is chosen. Furthermore, *ResourceCentric* terminates when no more assignments that do not conflict with the previously chosen assignments exist. As a result, *ResourceCentric* has to remove all assignments in $\mathscr{M}$ when it terminates, since otherwise it can at least add one of the remaining assignments to the chosen set. Moreover, once an assignment is removed from consideration at λ, it would not appear again in the later steps since it conflicts with m^λ. Hence, every term appearing on the right hand side also appears exactly once on the left so that the conclusion holds.

Finally, we establish the worst case ratio for *ResourceCentric* in the following theorem.

Theorem 3 *Applying ResourceCentric to the ST-MR-IA problem while restricting the maximum coalition size to be k yields a worst case ratio of $\theta = \min(2k+2, \max_{m_{jl} \in S^*}(|M_{jl}(1)|))$, in which S^* is an abbreviated notation for $S^*(R, T)$.*

Proof Let us first prove the $\max_{m_{jl} \in S^*}(|M_{jl}(1)|)$ part. At any greedy step λ, *ResourceCentric* needs to check all remaining assignments (i.e., assignments in (R_λ, T_λ), denoted by $\mathscr{M}(\lambda)$) and chooses the one with the maximum ρ value. We analyze the property of ρ by summing it over $\mathscr{M}(\lambda)$:

$$\begin{aligned} \sum_{m_{xy} \in \mathscr{M}(\lambda)} \rho_{xy} = & \sum_{m_{xy} \in \mathscr{M}(\lambda)} U(m_{xy}) \\ & - \sum_{m_{xy} \in \mathscr{M}(\lambda)} \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \end{aligned} \tag{10}$$

in which $\sum_{m_{xy} \in \mathscr{M}(\lambda)} U(m_{xy})$ is simply the sum of the utilities of all remaining assignments. In $\sum_{m_{xy} \in \mathscr{M}(\lambda)} \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})$, for any assignment $m_{pq} \in \mathscr{M}(\lambda)$, $U(m_{pq})$ appears only when $m_{xy} \in M_{pq}(\lambda)$. Hence, $U(m_{pq})$ appears $|M_{pq}(\lambda)|$ times and the total utility contributed to Eq. 10 is the negation of the following:

$$|M_{pq}(\lambda)| \cdot \frac{1}{|M_{pq}(\lambda)|} \cdot U(m_{pq}) = U(m_{pq}) \tag{11}$$

As Eq. 11 is true for every assignment in $\mathscr{M}(\lambda)$, we can conclude that $\sum_{m_{xy}\in\mathscr{M}(\lambda)} \rho_{xy} = 0$ in Eq. 10. As a result, in any greedy step, we can infer that at least one of the remaining assignments has a non-negative ρ value and consequently, the assignment that is chosen by *ResourceCentric* has a non-negative ρ value as it maximizes ρ. Based on this conclusion, we have $\forall\lambda$:

$$U(m^\lambda) \geq \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \tag{12}$$

The solution returned by *ResourceCentric* (denoted by $S^{RC}(R,T)$), is simply the summation of all greedy assignments in G:

$$\begin{aligned}
S^{RC}(R,T) &= \sum_{m^\lambda\in G} U(m^\lambda) \\
&\geq \sum_{m^\lambda\in G} \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) && \text{(Eq. 12)} \\
&\geq \sum_{m^\lambda\in G} \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(1)|} \cdot U(m_{jl}) && \text{(Lemma 1)} \\
&= \sum_{m_{jl}\in\mathscr{M}} \frac{1}{|M_{jl}(1)|} \cdot U(m_{jl}) && \text{(Lemma 2)} \\
&\geq \sum_{m_{jl}\in S^*} \frac{1}{|M_{jl}(1)|} \cdot U(m_{jl}) \\
&\geq \frac{1}{\max_{m_{jl}\in S^*}(|M_{jl}(1)|)} \cdot S^*(R,T)
\end{aligned} \tag{13}$$

Let us now prove the $2k+2$ part. At step λ, $M^*(\lambda)$ is used to denote the set of assignments of $\{m | m \in S^*, m \in \mathscr{M}(\lambda)$ and m conflicts with $m^\lambda\}$. As the size of any coalition is bounded by k, we have:

$$|M^*(\lambda)| \leq k+1 \tag{14}$$

Furthermore, according to the greedy criterion, we have $\forall\lambda\ \forall m_{pq} \in M^*(\lambda)$:

$$\begin{aligned}
&U(m^\lambda) - \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \\
&\geq U(m_{pq}) - \sum_{m_{jl}\in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})
\end{aligned} \tag{15}$$

Summing over all assignments in $M^*(\lambda)$ and then summing over all greedy steps λ on the right hand side, we have:

$$\begin{aligned}
&\sum_\lambda |M^*(\lambda)| \cdot (U(m^\lambda) - \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})) \\
&\geq \sum_\lambda \sum_{m_{pq}\in M^*(\lambda)} (U(m_{pq}) - \sum_{m_{jl}\in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}))
\end{aligned} \tag{16}$$

First of all, based on the definition of $M^*(\lambda)$, we know that any assignment in the optimal solution must appear exactly once as m_{pq} in $\sum_\lambda \sum_{m_{pq}\in M^*(\lambda)}$, since every assignment in the optimal solution must conflict with the chosen assignment at some step during the greedy process in order to be removed from further consideration (whether it is chosen as the greedy choice or not). Hence, it is straight forward to conclude that:

$$\sum_{\lambda} \sum_{m_{pq}\in M^*(\lambda)} U(m_{pq}) = S^*(R,T) \tag{17}$$

Furthermore, since any assignment can at most conflict with $k+1$ assignments in the optimal solution, we can conclude that any assignment can appear at most $k+1$ times as m_{jl} in $\sum_\lambda \sum_{m_{pq}\in M^*(\lambda)} \sum_{m_{jl}\in M_{pq}(\lambda)}$. Moreover, we know that any assignment conflicting with m^γ does not appear after m^γ is chosen at greedy step γ. To conclude from the above, an indicator function, ϕ, is introduced for comparing two assignments, which returns 1 only when the two are the same (0 otherwise). For any assignment m_{xy} that conflicts with m^γ, we compute its contribution to $\sum_\lambda \sum_{m_{pq}\in M^*(\lambda)} \sum_{m_{jl}\in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|}\cdot U(m_{jl})$ as follows:

$$\begin{aligned}
&\sum_{\lambda} \sum_{m_{pq}\in M^*(\lambda)} \sum_{m_{jl}\in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|}\cdot U(m_{jl})\cdot \phi(m_{jl},m_{xy}) \\
&= \sum_{\lambda\le\gamma} \sum_{m_{pq}\in M^*(\lambda)} \sum_{m_{jl}\in M_{pq}(\lambda)} \frac{1}{|M_{xy}(\lambda)|}\cdot U(m_{xy})\cdot \phi(m_{jl},m_{xy}) \\
&\le \sum_{\lambda\le\gamma} \sum_{m_{pq}\in M^*(\lambda)} \sum_{m_{jl}\in M_{pq}(\lambda)} \frac{1}{|M_{xy}(\gamma)|}\cdot U(m_{xy})\cdot \phi(m_{jl},m_{xy}) \qquad \text{(Lemma 1)} \\
&= \frac{1}{|M_{xy}(\gamma)|}\cdot U(m_{xy})\cdot \sum_{\lambda\le\gamma} \sum_{m_{pq}\in M^*(\lambda)} \sum_{m_{jl}\in M_{pq}(\lambda)} \phi(m_{jl},m_{xy}) \\
&\le (k+1)\cdot \frac{1}{|M_{xy}(\gamma)|}\cdot U(m_{xy})
\end{aligned} \tag{18}$$

Summing over all assignments removed at step γ and then summing over all steps γ on the right hand side (in such a way, all assignments in $\mathscr{M}$ are included exactly once, so that we can remove the indicator function on the left) gives:

$$\begin{aligned}
&\sum_{\lambda} \sum_{m_{pq}\in M^*(\lambda)} \sum_{m_{jl}\in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|}\cdot U(m_{jl}) \\
&\le (k+1)\cdot \sum_{\gamma} \sum_{m_{xy}\in M^\gamma(\gamma)} \frac{1}{|M_{xy}(\gamma)|}\cdot U(m_{xy}) \\
&= (k+1)\cdot \sum_{\lambda} \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|}\cdot U(m_{jl})
\end{aligned} \tag{19}$$

Incorporating equations 17 and 19 into Eq. 16, we have:

$$\begin{aligned}
&\sum_{\lambda} |M^*(\lambda)|\cdot \Big(U(m^\lambda) - \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|}\cdot U(m_{jl})\Big) \\
&\ge S^*(R,T) - (k+1)\cdot \sum_{\lambda} \sum_{m_{jl}\in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|}\cdot U(m_{jl})
\end{aligned} \tag{20}$$

Given that $|M^*(\lambda)| \geq 0$ and Eq. 14, it can also be concluded that:

$$\begin{aligned}
&\sum_{\lambda} |M^*(\lambda)| \cdot (U(m^{\lambda}) - \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})) \\
&\leq \sum_{\lambda} |M^*(\lambda)| \cdot U(m^{\lambda}) \\
&\leq (k+1) \cdot S^{RC}(R,T) \qquad (21)
\end{aligned}$$

Combining equations 20 and 21, we have:

$$\begin{aligned}
&(k+1) \cdot S^{RC}(R,T) \\
&\geq S^*(R,T) - (k+1) \cdot \sum_{\lambda} \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \\
&\geq S^*(R,T) - (k+1) \cdot S^{RC}(R,T) \qquad \text{(First inequality in Eq. 13)}
\end{aligned}$$

And finally, it can be concluded that:

$$S^*(R,T) \leq (2k+2) \cdot S^{RC}(R,T) \qquad (22)$$

Hence the conclusion holds.

For problem instances in which assignments in the optimal solution conflict less with other assignments, the $\max_{m_{jl} \in S^*}(|M_{jl}(1)|)$ ratio guarantees a good quality solution. This is especially true in multi-robot systems with heterogeneous robots, since different robots can only handle specific tasks. Otherwise, the solution quality is bounded by $2k+2$. Although this worst case ratio is slightly worse than *AverageUtility*, it is more difficult to satisfy the boundary conditions (i.e., all inequalities in the proof hold as equalities simultaneously), which may suggest that *ResourceCentric* would perform well on average even when $\max_{m_{jl} \in S^*}(|M_{jl}(1)|)$ is large.

5.4 The algorithm of *ResourceCentric*

Algorithm 1 presents an implementation of *ResourceCentric* using a graph structure. The algorithm starts with building the graph, which has a node for each assignment and an edge between two nodes if they conflict with each other. Then the algorithm proceeds with the greedy iterations and assigns a task at each step. For every assignment made, the assignment node and all assignments connecting with it, as well as all connecting edges,[11] are removed from the graph. This process continues until all assignments are removed (i.e., the graph becomes empty).

The complexity for creating the graph is bounded by $O(|T||C||\mathscr{M}|)$. Each greedy step is bounded by $O(|\mathscr{M}|^2)$.[12] As there can at most be $\min(|R|, |T|)$ assignments, the complexity for the entire process is bounded by $O(\min(|R|, |T|) \cdot |T|^2|C|^2)$ (note that $|\mathscr{M}|$ is bounded by $|T||C|$).

[11] In our implementation of *ResourceCentric*, we do not keep the edge information in the node structure in order to reduce the space complexity.

[12] Note that computing $\frac{1}{|M_{il}|}$ in the inner loop requires only constant time in our algorithm. For each node, this count is computed when the graph is initially created and kept updated when a chosen assignment and its neighbors are removed from the graph.

Algorithm 1 *ResourceCentric*

Generate the set of coalitions C, with maximum size k.
Create an undirected graph $G : (V, E)$.
for all t_l in T **do**
  for all c_j in C **do**
    if c_j satisfies t_l **then**
      Create a node m_{jl}.
      Compute $U(m_{jl})$.
      for all $v \in V$ **do**
        if v conflicts with m_{jl} **then**
          Connect v and m_{jl}.
        end if
      end for
    end if
  end for
end for
while G is not empty **do**
  for all v in V **do**
    for all u: u and v are connected **do**
      Compute $\frac{1}{|M_u|} \cdot U(u)$
    end for
    Compute ρ_v.
  end for
  Choose v^{RC} that maximizes ρ.
  Remove v^{RC}, its neighbors and edges connecting these nodes with the remaining nodes from G.
end while
return The chosen nodes (i.e., assignments).

5.5 ResourceCentricApprox

One problem with *ResourceCentric*, however, is the computational complexity. Although *ResourceCentric* runs in polynomial time with respect to $|C|$, as we discussed, $|C|$ can grow exponentially with $|R|$. When the number of robots in the distributed system is large, *ResourceCentric* can be significantly slower than *AverageUtility* and *MaxUtility*, which run in $O(\min(|R|, |T|) \cdot |T||C|)$. Instead of computing ρ exactly, we can use the following approximation at greedy step λ:

$$
\begin{aligned}
\hat{\rho}_{xy} &= U(m_{xy}) - \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \\
&\approx U(m_{xy}) - \sum_{r_i \in c_x} \sum_{m_{jl} \in M_i(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) && \text{(Break onto each robot)} \\
&= U(m_{xy}) - \sum_{r_i \in c_x} \overline{\frac{|M_i(\lambda)|}{|M_{jl}(\lambda)|} \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} \\
&\approx U(m_{xy}) - \sum_{r_i \in c_x} \overline{\frac{|M_{i,l}(\lambda)|}{|M_l(\lambda)|} \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} && \text{(Remove dependency on } j\text{)} \\
&= U(m_{xy}) - \sum_{r_i \in c_x} \overline{\theta_{il}(\lambda) \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} && (23)
\end{aligned}
$$

in which the bars over the formulas represent averaging operations, $M_i(\lambda)$ represents the set of assignments in (R_λ, T_λ) that rely on r_i (i.e., r_i is in the coalitions for the assignments),

Table 1 Summary of discussed methods with maximum coalition size k

Name	Formulation	Worst case ratio	Amort. worst case time
AverageUtility	Theorem 1	$2k$	$O(\min(\|R\|, \|T\|) \cdot \|T\|\|C\|)$
MaxUtility	Theorem 2	$k+1$	$O(\min(\|R\|, \|T\|) \cdot \|T\|\|C\|)$
ResourceCentric	Eq. 8	$\min(2k+2, \max_{m_{jl} \in S^*}(\|M_{jl}(1)\|))$	$O(\min(\|R\|, \|T\|) \cdot \|T\|^2\|C\|^2)$
ResourceCentricApprox	Eq. 23	Not determined	$O(\min(\|R\|, \|T\|) \cdot \|R\|\|T\|^2\|C\|)$

$M_{i,l}(\lambda)$ represents the set of assignments in (R_λ, T_λ) for task t_l relying on r_i, and $M_l(\lambda)$ represents the set of assignments in (R_λ, T_λ) for task t_l. The $\approx$ sign in the above formula should be interpreted as "is approximated by". Note that in the first approximation, the same assignments may appear more than once in $\sum_{r_i \in c_x} \sum_{m_{jl} \in M_i(\lambda)}$. The second approximation is introduced to avoid nested loops for iterating through all coalitions (each loop is on the order of $O(|C|)$), so that the computational complexity can be reduced.

At each step λ, for each remaining task t_l, we compute $\theta_{il}(\lambda)$ for each remaining r_i, which is a measure that reflects how much t_l relies on r_i. When choosing an assignment m_{xy}, for each robot $r_i \in c_x$, we first compute the expected loss of utility due to the assignment of r_i as:

$$E_\lambda(r_i) = \overline{\theta_{il}(\lambda) \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} \tag{24}$$

Afterwards, we compute $\hat{\rho}_{xy} = U(m_{xy}) - \sum_{r_i \in c_x} E_\lambda(r_i)$ and choose the assignment that maximizes it. This heuristic is referred to as *ResourceCentricApprox*.

Now let's trace back to our motivational example at the beginning of this section. Suppose that the only three robots that have the first three capabilities are r_1, r_2, r_3 with capability vectors $\boldsymbol{B}_1 = (1,0,0,0,0)$, $\boldsymbol{B}_2 = (0,1,0,0,0)$, $\boldsymbol{B}_3 = (0,0,1,0,0)$. In this case, we have $\theta_{12} = \theta_{23} = \theta_{34} = 1.0$, since robot r_1, r_2 and r_3 are prerequisites for tasks t_2, t_3 and t_4, respectively. As a result, when assigning t_1 with a slightly higher reward, the value of $\hat{\rho}$ for the assignment of $\{r_1, r_2, r_3\} \rightarrow t_1$ would still be lower than for the other tasks and hence t_1 would not be chosen. Although this may seem to be an arbitrary example, we show in the result sections that *ResourceCentric* and *ResourceCentricApprox* actually perform better, which indicates that similar situations often occur in random configurations.

5.6 The algorithm of *ResourceCentricApprox*

One way to implement *ResourceCentricApprox* is presented in Algorithm 2. The algorithm starts with creating a hash table for each task and a hash table for each robot and task pair. After it fills the hash tables, the algorithm proceeds with the greedy iterations to make assignments.

The complexity for creating the hash tables is bounded by $O(|R||T|)$. The complexity for filling the hash tables is $O(|T||C|)$. Each greedy choice requires $O(|R||T|^2|C|)$ computations in the worst case. Hence, the computational complexity for this implementation is bounded by $O(\min(|R|, |T|) \cdot |R||T|^2|C|)$. Hence, we can conclude that this algorithm performs almost as well as *AverageUtility* and *MaxUtility* in terms of worst case running time.

Table 1 provides a summary of all methods we have discussed in this paper, including their formulations, worst case ratios, and amortized worst case running times.

Algorithm 2 *ResourceCentricApprox*

Generate the set of coalitions C, with maximum size k
for all t_l in T **do**
  Create a hash table H_l
  for all i in R **do**
    Create a hash table $H_{i,l}$
  end for
end for
for all t_l in T **do**
  for all c_j in C **do**
    if c_j satisfies t_l **then**
      Add c_j into H_l
      for all r_i in c_j **do**
        Add c_j into $H_{i,l}$.
      end for
    end if
  end for
end for
while H tables are not all empty **do**
  for all r_i in remaining R **do**
    for all t_l in remaining T **do**
      for all c_j: $r_i \in c_j$ **do**
        Compute $\frac{|H_{i,l}|}{|H_l|} \cdot U(m_{jl})$.
      end for
    end for
    Compute $E(r_i)$.
  end for
  for all m_{xy} remaining **do**
    Compute $\hat{\rho}_{xy} = U(m_{xy}) - \sum_{r_i \in c_x} E(r_i)$.
  end for
  Choose m^{RCA} that maximizes $\hat{\rho}$.
  for all r_i in c^{RCA} **do**
    for all t_l in T **do**
      for all c in $H_{i,l}$ **do**
        Remove c from all tables.
      end for
    end for
  end for
  Clear tables involving t^{RCA}.
end while

6 Extended formulation

One issue with the formulation of ST-MR-IA is that it is insufficient for complicated scenarios. Hence, in this section, we extend the formulation of the ST-MR-IA problem to incorporate general task dependencies. This extended formulation is referred to as the ST-MR-IA-TD problem. A result on the hardness of approximating ST-MR-IA-TD is provided afterwards. An algorithm that utilizes the discussed methods for ST-MR-IA to address this extended formulation of the problem is also provided.

6.1 Adding task dependencies

As discussed, one issue with the formulation of ST-MR-IA is that it does not incorporate task dependencies, which can be critical for real world applications. Previous approaches (e.g., [18]) have studied precedence ordering between tasks, in which one task can only be

assigned when other tasks in its precedence order are also assigned. For example, in the disaster response scenario we presented earlier, the task for addressing the fires in buildings is not possible if the roads to the buildings are not cleared. However, we realize that the definition of precedence order should be extended to incorporate scenarios in which assigning other tasks facilitates the execution of one task, instead of being a prerequisite for the task. For example, although there might exist an alternative road that is not blocked, taking this alternative route can potentially be less optimal than clearing the blocked road first and using it to reach the buildings (e.g., it may take longer to reach the buildings via the alternative route). Note that the precedence order in [18] can then be considered as a special case of this extended formulation (i.e., a task yields a very small utility if the tasks in its precedence order are not satisfied, essentially making the task be ignored until its precedence order is satisfied). Also note that the precedence order does not necessarily have to specify a complete ordering of task execution (i.e., some tasks have to be completed before starting the execution of some others). A task and tasks in its precedence order may or may not be able to be executed simultaneously. To avoid the scheduling issue that may arise from this complexity, in this work, we follow a similar approach as in [18], such that we pre-allocate resources to the task and tasks in its precedence order at the same time.

Another aspect in task dependencies that has not been considered occurs when there are alternative tasks, such that the assignment of any one of them makes the others unnecessary. In the same example, when there are alternative roads that are all blocked leading to the same buildings, only one of them needs to be cleared. Correspondingly, these other tasks may still be beneficial even though one of them is assigned. For example, to achieve more efficiency, several alternative roads may need to be cleared (i.e., for other traffic coming in and out without interfering with the truck agents). To incorporate these considerations, we add the following component into the formulation of ST-MR-IA:

– a set of task dependencies Γ. Each dependency for a task t is defined as a pair $\tau = (\mathscr{T}, \Re^+)$, in which $\mathscr{T} \subseteq \{T - t\}$ and $\mathscr{T} \neq \emptyset$. The real positive value (denoted by v_D) is the updated reward of t when this dependency is satisfied.

For example, for specifying a linear ordering between t_1, t_2 and t_3 such that t_1 must be satisfied before t_2 and t_2 must be satisfied before t_3, we need to define a task dependency $(\{t_1\}, v_D)$ for t_2 and a task dependency $(\{t_2\}, v'_D)$ for t_3. We denote this extended formulation of the ST-MR-IA problem as ST-MR-IA-TD. Given a task t_l, precedence orders can then be implemented by requiring that $v_D \geq \boldsymbol{V}[l]$; alternative tasks can be implemented by requiring that $v_D < \boldsymbol{V}[l]$. Dependencies of these two aspects are considered separately in the algorithm we present at the end of this section. This is due to the fact that when $v_D \geq \boldsymbol{V}[l]$, tasks in the dependency are desirable; on the other hand, when $v_D < \boldsymbol{V}[l]$, tasks in the dependency should generally be avoided (when v_D for the task is so small such that the utility for any assignment is non-positive, the task would effectively be ignored). In cases when a task has multiple dependencies, rules should be defined for cases when multiple dependencies are satisfied simultaneously. This aspect will be addressed in more depth in our future work.

6.2 Problem analysis

In this section, we provide the result on the hardness of approximating the ST-MR-IA-TD problem.

Theorem 4 *It is NP-hard to approximate the ST-MR-IA-TD problem with a poly-time worst case ratio that is independent of v_D values in task dependencies.*[13]

Proof The proof is by contradiction. Let's suppose that a polynomial time approximation algorithm, *TD_Approx*, does exist with a poly-time worst case ratio of θ. Next, we show that we can utilize algorithm *TD_Approx* to solve the *3-Partition* problem, which is strongly NP-complete. Any instance of the *3-Partition* problem can be represented as *3-Partition*$(S, M : S = \{e_1, e_2, \ldots, e_{3M}\}, e_j \in \mathbb{Z}^+)$. Let the sum of all elements in S be $M \cdot B$. The problem is to determine whether or not the set S can be divided into M sets such that the sum of the elements in each set is B. The problem remains NP-complete even when we require $\forall e_j \in S : \frac{B}{4} < e_j < \frac{B}{2}$. These constraints imply that if a solution exists for the *3-Partition* problem, each set would have exactly 3 elements.

Next, we show how we construct an instance of *ST-MR-IA-TD(R, C, T,* $\boldsymbol{W}$*,* $\boldsymbol{V}$*, Cost,* Γ*)* from an instance of *3-Partition*(S, M). First of all, for each element $e_j \in S$, we construct a robot r_j which has a *1-D* (i.e., $H = 1$) capability vector with value equal to the integer value of the element e_j. Then, we create the set C of coalitions by including all coalitions with exactly 3 robots. Note the size of $|C|$ is $\binom{3M}{3}$, which is polynomial in M. This is important for guaranteeing the validity of the transformation. On the other hand, this does not influence the determination of the existence of a solution, as each set would have exactly 3 elements anyway.

We create M tasks to be accomplished, so that $T = \{t_1, t_2, \ldots, t_M\}$. Each t_l has a capability requirement of exactly B. Since $H = 1$, the cost vector $\boldsymbol{W}$ reduces to a scalar w, which we set to 0 for simplicity. The initial rewards for all tasks are assigned to be 1. We assume that there is no communication cost so that the function $Cost$ invariably returns 0. Finally, for Γ, we define a task dependency, $(\{T - t_M\}, v_D)$, for t_M. Since θ is assumed to be computable in polynomial time and independent of v_D values, we compute θ and assign v_D to be $M \cdot \theta - (M - 1)$. We have thus constructed an instance of the ST-MR-IA-TD problem from an instance of the *3-Partition* problem.

Now the *3-Partition* problem can be solved using *TD_Approx* as follows.

$$3\text{-}Partition = \begin{cases} 1 \textbf{ if } TD\_Approx \geq M \\ 0 \textbf{ if } otherwise \end{cases} \tag{25}$$

Whenever *TD_Approx* returns an overall utility of no less than M, we know that all M tasks must be in the solution. This is because accomplishing less than M would receive an overall utility no more than $M - 1$. Hence, we know that there exists a way in the ST-MR-IA-TD problem to allocate exactly 3 robots to each task. This solution is clearly also a solution for the *3-Partition* problem. On the other hand, if *TD_Approx* returns an overall utility of less than M, we know that the optimal solution must achieve an overall utility of less than $M \cdot \theta$, according to the definition of worst case ratio. If there exists a solution for the *3-Partition* problem, we can apply the solution to the ST-MR-IA-TD problem so that all tasks are included. The corresponding overall utility received is then $M \cdot \theta - (M - 1) + (M - 1) = M \cdot \theta$. This contradicts with the previous conclusion that the optimal solution must achieve an overall utility of no more than $M \cdot \theta$. Hence, there could not exist a solution for the *3-Partition* problem. In such a way, we have solved the *3-Partition* problem. Unless $P \equiv NP$, we have proven that the conclusion holds. □

[13] Recall that a poly-time worst case ratio is a worst case ratio that can be computed in polynomial time given the problem instance.

6.3 Allocation with task dependencies

For addressing the ST-MR-IA-TD problem, we adapt the greedy approach used in [18] for task allocation with precedence orders. However, since task dependencies can either increase or decrease the reward, we consider the two cases separately. For any task t_l with a set of dependencies $\Gamma_l \subseteq \Gamma$, we separate Γ_l into two disjoint sets:

- Γ^p: the set of dependencies that satisfies $v_D \geq \boldsymbol{V}[l]$.
- Γ^r: the set of dependencies that satisfies $v_D < \boldsymbol{V}[l]$.

We consider Γ^p as the set of precedence orders in [18]. As in [18], at each greedy step, instead of making a single assignment at a time, the algorithm checks each task with tasks in each of its precedence orders (i.e., dependencies in Γ^p for the task). For each set of tasks, a set of assignments are made using one of the previously discussed methods for the ST-MR-IA problem. The algorithm then chooses the set of assignments that gives the best *quality measure*[14] (similar to p-value in [18]). The difference from [18] is that task dependencies in Γ^r are also considered in the computation of the *quality measure* for each set of assignments to incorporate their influences. Note that although we cannot directly compare our results with [18] due to the different measures used (i.e., cost measures in [18] rather than utility measures in our work) and that we also consider Γ^r, the algorithm in Sect. 6.4 using *AverageUtility* is in fact very similar to that in [18] when ignoring these differences.

To simplify the rules when multiple dependencies are satisfied simultaneously, for each task t_l, we let all dependencies in Γ^p assume the same value v_D^p and all dependencies in Γ^r assume v_D^r. When only dependencies in Γ^p are satisfied, the reward of the task is updated to be v_D^p; when only dependencies in Γ^r are satisfied, the reward is updated to be v_D^r; when there are satisfied dependencies in both sets, the reward is updated to be v_D^r. This set of rules is reasonable when tasks tend to have few dependencies (i.e., at most 1 in either set) or dependencies from the same set have similar effects on the task. More complicated rules can be designed without influencing the following discussions. The influences of task dependencies on the task rewards are computed according to these rules.

6.4 The algorithm for task allocation with task dependencies

The algorithm to address the ST-MR-IA-TD problem is shown in Algorithm 3. At every step, for every remaining task, for every task dependency in Γ^p for the task, the algorithm creates a set of tasks that includes the task along with all other tasks in the dependency. This set of tasks is fed to one of the methods for addressing the ST-MR-IA problem. After the assignments are made, the *quality measure* for this set of assignments is evaluated. Note that all dependencies (i.e., both Γ^p and Γ^r) of the previously chosen tasks and tasks in these assignments are checked to compute the influence (i.e., due to the updates of task rewards) on the *quality measure* due to newly satisfied dependencies. The set of assignments with the best *quality measure* is then chosen.

[14] The *quality measure* for a set of assignments is computed as the combination of the measures, used by the chosen method for making these assignments (e.g., ρ for *ResourceCentric*), while incorporating the influences of task dependencies. For example, when using *MaxUtility*, the *quality measure* is computed as the summation of the utility measures for these assignments, considering the change of task rewards due to the satisfied task dependencies.

Algorithm 3 Task allocation with task dependencies

while remaining tasks can still be assigned **do**
  for all t_l in T **do**
    for all τ in Γ^p for t_l **do**
      Create the set of tasks to include t_l and tasks in τ.
      Invoke a method (e.g., *AverageUtility,* ...) to choose assignments for this set of tasks.
      Record the chosen assignments as $M_{l\tau}$.
      if $M_{l\tau} \neq \emptyset$ **then**
        for all $m_{jl} \in M_{l\tau}$ **do**
          Compute the measure for the greedy choice based on the chosen method.
          Compute the influence of newly satisfied dependencies (based on Γ^p and Γ^r of tasks in the chosen assignments, including t_l) as a result of choosing m_{jl}.
          Incorporate the influence into the measure for the greedy choice.
          Assume that m_{jl} is chosen for the next iteration.
        end for
        Combine the measures for all $m_{jl} \in M_{l\tau}$ as the *quality measure*.
      end if
    end for
  end for
  Choose the $M^*_{l\tau}$ with the maximum *quality measure*.
  Remove the assigned robots and tasks in $M^*_{l\tau}$ from R and T.
end while

7 Simulation results for ST-MR-IA

In this section, we provide simulation results for ST-MR-IA. We first illustrate cases when the natural heuristics can produce poor quality solutions. Afterwards, we compare the performance of the natural heuristics with *ResourceCentric* and *ResourceCentricApprox* in random configurations. Then, simulation results with different robot capability levels for tasks are provided. Finally, we present results with varying maximum coalitions sizes and results with a random cost function for communication and coordination. In all simulations except for the first one, or when specified otherwise, the costs of capabilities (i.e., $\boldsymbol{W}$) are randomly generated from [0.0, 1.0]; each robot or task has a 50 % chance to have or require any capability and the capability values are randomly generated from $[0, 8]$; task rewards are randomly generated from $[100, 200]$ and $Cost$ is assumed to be a linear function of the number of robots (i.e., $4n$). All statistics are collected over 100 runs.

7.1 Comparison with limited capability resources

Based on the previous discussions, we can see that the limitation of capability resources is the key influential factor causing *AverageUtility* and *MaxUtility* to produce bad solutions. Hence in this simulation, we specifically create two types of tasks and define two limited capabilities. The first type of task requires both limited capabilities and has a slightly higher reward, while the other task type requires only one of the limited capabilities. Beside the limited capabilities, we also define two common capabilities that many robots have and most of the tasks require. It is also assumed that every robot has only one capability. For each of the two limited capabilities, we create only two robots with that capability while varying the number of robots with common capabilities. A sufficient number of tasks are generated for both types. The maximum size of the coalitions is restricted to be 3 in this simulation.

Figure 1 shows the results. While Fig. 1a shows the average performance ratios (i.e., compared to the optimal solution), Fig. 1b shows the average performance ratios with stan-

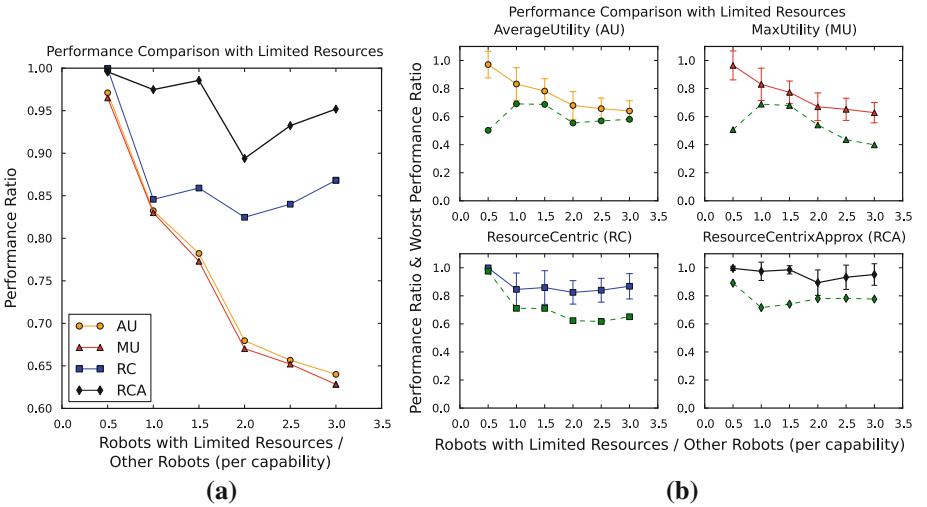

Fig. 1 Task allocation with limited capability resources. **a** Average performance ratios. **b** Separate and more detailed results with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

dard deviations and the worst performance ratios out of all 100 runs, separately for all four methods. We can see that as the number of robots with common capabilities increases (so that more tasks can be assigned), the ratios of *AverageUtility* and *MaxUtility* decrease drastically. This is because both heuristics tend to choose tasks of the first type with higher rewards, although these tasks consume more of the limited capabilities such that other tasks can be significantly influenced. *ResourceCentric* and *ResourceCentricApprox*, on the other hand, consider the influence of the consumption of these limited capabilities when choosing assignments. From Fig. 1, we can see that the performances of *AverageUtility* and *MaxUtility* keep decreasing (i.e., to around 60 %) as the number of robots with common capabilities increases. Another note is that *ResourceCentricApprox* performs better than *ResourceCentric*. This is due to the fact that the measure for the greedy choice in *ResourceCentricApprox* (i.e., $\hat{\rho}$) directly optimizes on the limited capabilities (i.e., robots) in this simulation.

7.2 Comparison with random configurations

However, we are more interested in performances in random configurations. In this simulation, we increase the number of capabilities to 7. In the remainder of this and the next sections, unless specified otherwise, the maximum coalition size is set to be 5. We vary the number of robots while fixing the number of tasks. Figure 2 shows the results. Table 2 shows the outcome from t-tests that are run to determine the statistical significance of the results in Fig. 2. For each pair of the discussed methods, for each data point in the figure, we use a 'y' (yes) or 'n' (no) to indicate whether the results of the two methods being compared are significantly different. We can see good performances for *AverageUtility* and *MaxUtility*. However, *ResourceCentric* and *ResourceCentricApprox* still perform better. Another observation is that the new methods almost always have smaller standard deviations.

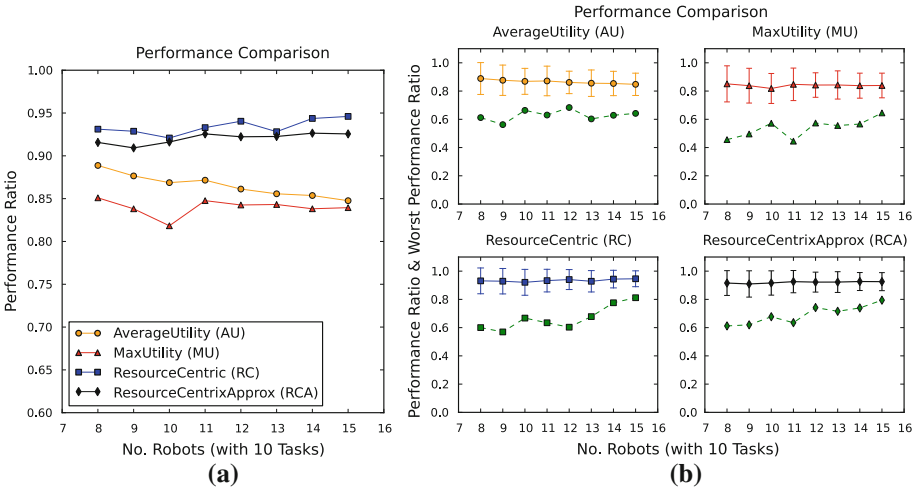

Fig. 2 Task allocation with random configurations. **a** Average performance ratios. **b** Separate and more detailed results with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

Table 2 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 2 (left to right) with $\alpha = 0.05$

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyyyyy	yyyyyyyy	yyyyyyyy	yyyyyyyy	nnnnynyy	yyynnnnn

Second row indicates, for each pair of methods, for each data point, whether the results are significantly different (see text for more explanation)

Table 3 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 3a (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyyyyy	yyyyyyyy	yyyyyyyy	ynnyyyyy	ynynyyyy	nnnnnnnn

Table 4 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 3b (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyyyyy	yyyyyyyy	yyyyyyyy	yyyyyyyy	nnnynyyy	nnyyyynn

7.3 Comparison with different robot capability levels

In this simulation, we present results for random configurations with robots of different capability levels compared to the tasks (i.e., determined by the maximum values for randomly generating the capability values). Figure 3a and shows the results for less and more capable robots, with maximum values of 4 and 12 respectively, while the results for the statistical significance are shown in Tables 3 and 4. We again vary the number of robots while fixing the number of tasks. These results show that *ResourceCentric* performs the best in all cases, although not always significantly different from *ResourceCentricApprox*.

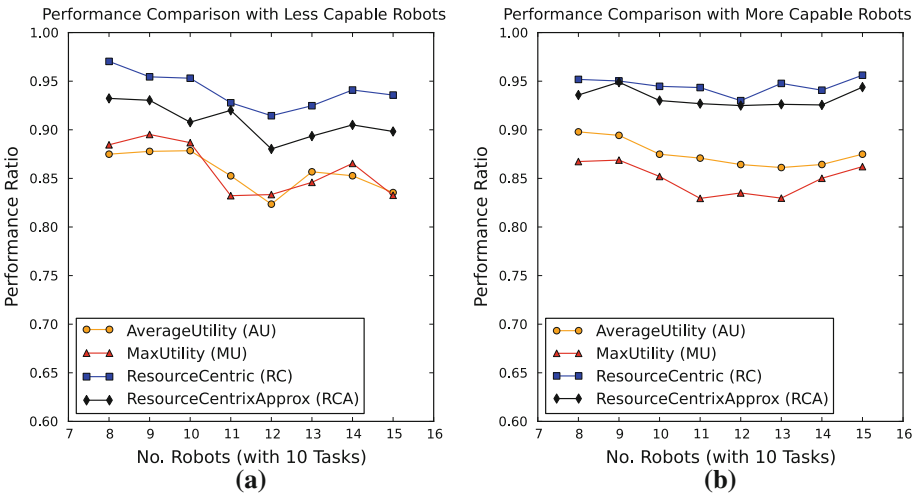

Fig. 3 Task allocation with different robot capability levels. **a** Average performance ratios with less capable robots for tasks. **b** Average performance ratios with more capable robots for tasks

Table 5 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 4 (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyy	yyyyy	yyyyy	yyyyy	yynnn	nnynn

7.4 Comparison with varying coalition sizes

In this simulation, we vary the maximum size of the coalitions from 3 to 11 while keeping all other settings similar to the previous simulations. Figure 4 and Table 5 show the results, which illustrate similar conclusions. While *ResourceCentric* and *ResourceCentricApprox* still perform significantly better than the other two methods, *ResourceCentric* performs only slightly better than *ResourceCentricApprox*.

7.5 Comparison with random *Cost* function

In this simulation, we investigate the influence of the *Cost* function. Instead of defining the communication and coordination cost to be linear in the number of robots in the coalition (i.e., $4n$), *Cost* returns a random value from $[0, 4n]$. Figure 5 and Table 6 present the results. While the conclusions regarding *ResourceCentric* and *ResourceCentricApprox* do not change, one obvious difference from the previous simulations is that the performance of *MaxUtility* significantly drops. This shows that *MaxUtility* is more sensitive to the change of the *Cost* function than the other methods. For the previous *Cost* function, a coalition with more robots is less likely to be chosen by all methods. However, when *Cost* returns a random number, *MaxUtility* cannot recognize that a coalition with fewer robots is often a better choice. For example, suppose that a task t_1 can be accomplished by $\{r_1, r_2\}$. As a result, $\{r_1, r_2, r_3\}$ can also accomplish the task. When the *Cost* function is linear in the number of robots in the coalitions, the coalition of $\{r_1, r_2\}$ would always be chosen by *MaxUtility*.

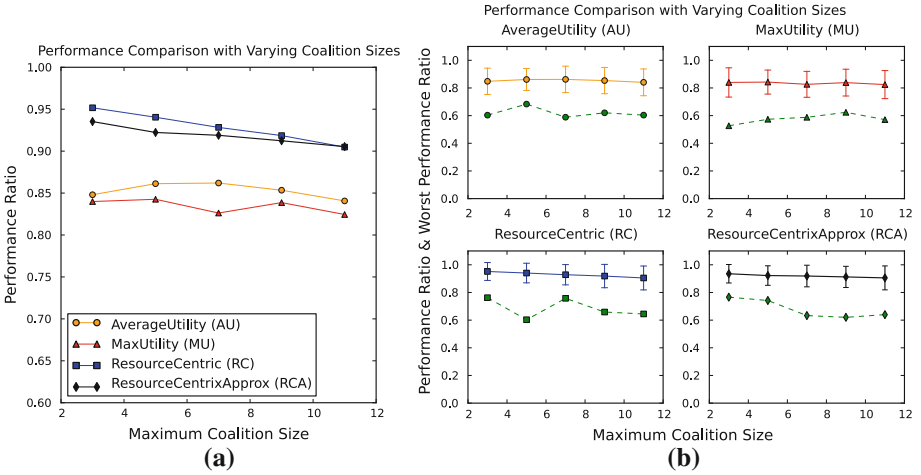

Fig. 4 Task allocation with varying coalition sizes. **a** Average performance ratios. **b** Separate and more detailed results with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

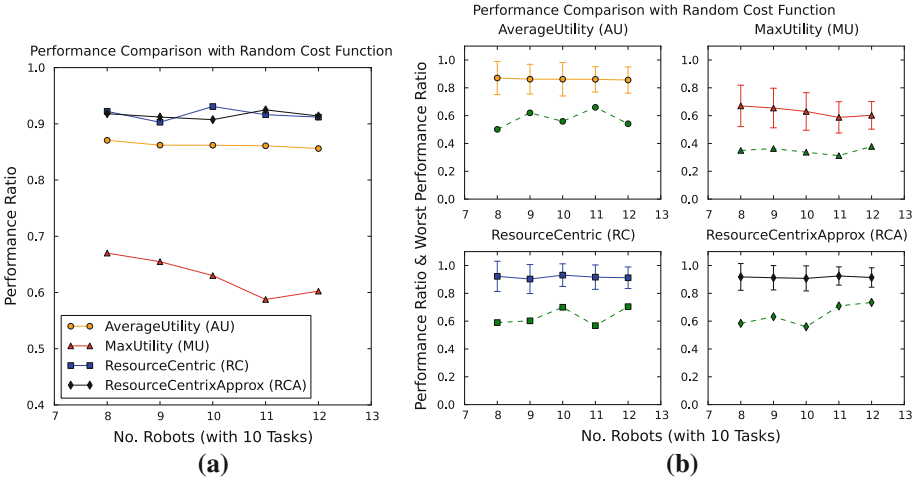

Fig. 5 Task allocation with a random cost function. **a** Average performance ratios. **b** Separate and more detailed results with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

However, when the function is random, *MaxUtility* cannot identify that $\{r_1, r_2\}$ may often be a better choice, since r_3 is then made available to other tasks.

7.6 Key findings from ST-MR-IA results

In this section, we have provided simulation results for comparing the performances of the previously discussed methods for addressing the ST-MR-IA problem. First of all, we provide simple scenarios in which *AverageUtility* and *MaxUtility* can perform badly. Furthermore, we show that *ResourceCentric* and *ResourceCentricApprox*, while considering inter-task

Table 6 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 5 (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyy	yyyyy	yyyyy	yyyyy	nnynn	yyyyy

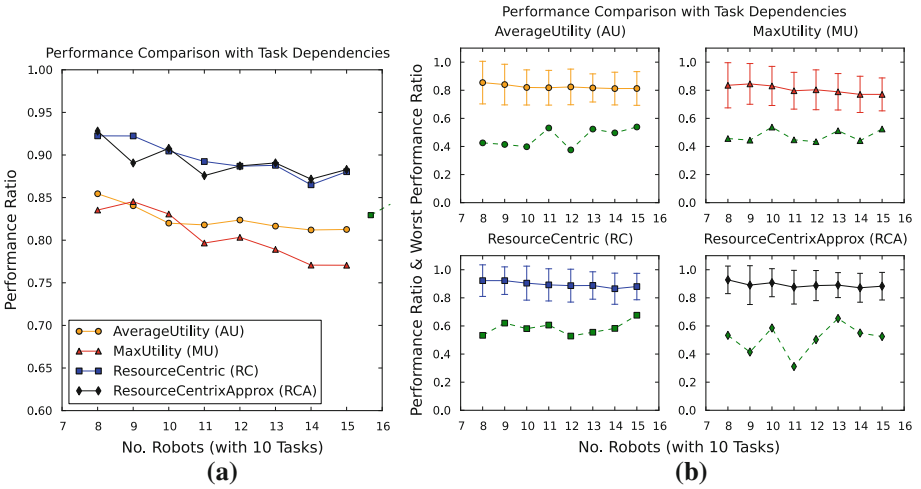

Fig. 6 Task allocation with task dependencies with random configurations. **a** Average performance ratios. **b** Separate and more detailed average performance ratios with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

resource constraints, not only perform better in these special scenarios, but also in random configurations. This suggests that these constraints are indeed commonly present in arbitrary configurations. Moreover, statistical testing shows that *ResourceCentric* and *ResourceCentricApprox* perform better than the other two methods with significant differences.

8 Simulation results for ST-MR-IA-TD

To generate dependencies for each task, in these simulations, we assume that the numbers of task dependencies in Γ^p and Γ^r are randomly chosen from {0, 1, 2}. Furthermore, for each task dependency, every other task has a probability of 0.2 to be included. Unless specified otherwise, v_D^r values for tasks are randomly generated from [0, 100] and v_D^p values are from [200, 400]. After presenting simulation results with random configurations, we show results with varying maximum coalition sizes. Results with a random cost function for communication and coordination are presented afterwards. Finally, results illustrating the influence of v_D are provided and time analyses for all methods are given.

8.1 Task dependencies with random configurations

First of all, we show results for the ST-MR-IA-TD problem with random configurations in Fig. 6 and Table 7. Compared to the performance ratios of simulation results for the ST-MR-IA problem, the performance ratios are slightly worse (approximately 5 % lower),

Table 7 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 6 (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyyyyy	yyyyyyyy	yyyyyyyy	yyyyyyyy	nynnnnnn	nnnnnnyy

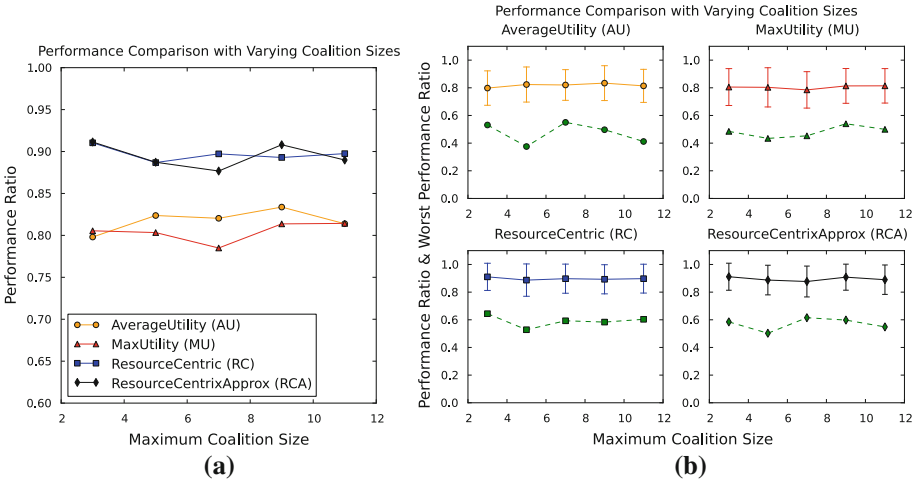

Fig. 7 Task allocation with task dependences with varying coalition sizes. **a** Average performance ratios. **b** Separate and more detailed average performance ratios with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

Table 8 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 7 (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyy	yyyyy	yyyyy	yyyyy	nnnnn	nnynn

which reveals that the ST-MR-IA-TD problem is indeed more difficult. Furthermore, the performances gradually decrease for all methods as the number of robots increases, such that more tasks are assigned and the influence of task dependencies becomes more prominent. Otherwise, we can still see that *ResourceCentric* and *ResourceCentricApprox* perform better than *AverageUtility* and *MaxUtility*.

8.2 Task dependencies with varying coalition sizes

Next, we show comparison results with varying maximum coalition sizes in Fig. 7 and Table 8. All methods perform similarly as for the ST-MR-IA problem (Fig. 4), although their performances also decrease slightly for the new formulation of the problem. Again, we can see that the maximum coalition size does not influence the performance very much.

8.3 Task dependencies with random *Cost* function

In this simulation, the *Cost* function is defined similarly as in the corresponding simulation for the ST-MR-IA problem. The results are shown in Fig. 8 and Table 9. Again, we can see

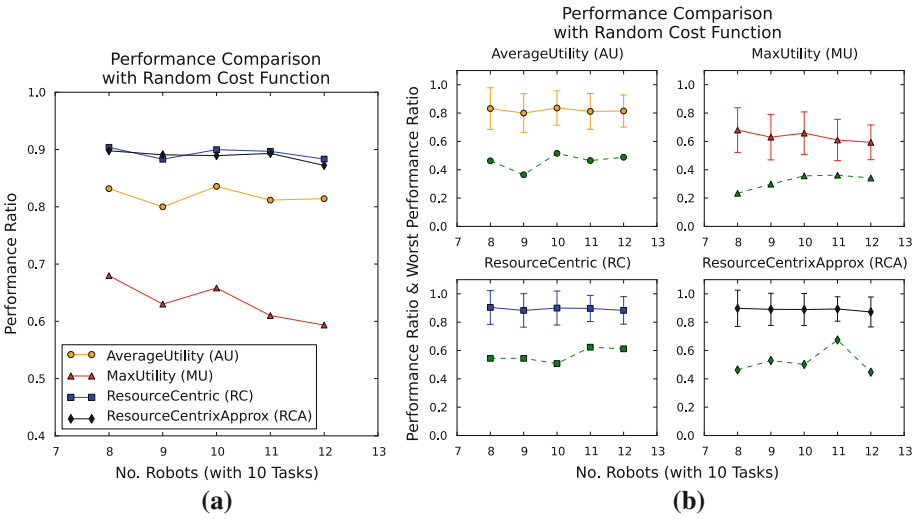

Fig. 8 Task allocation with task dependences with a random cost function. **a** Average performance ratios. **b** Separate and more detailed average performance ratios with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

Table 9 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 8 (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyy	yyyyy	yyyyy	yyyyy	nnnnn	yyyyy

that *MaxUtility* is the most sensitive to the change of the $Cost$ function. *ResourceCentric* and *ResourceCentricApprox* still perform better than *AverageUtility* and *MaxUtility* with significant differences, while *AverageUtility* performs better than *MaxUtility* with significant differences.

8.4 Varying maximum v_D^p values of task dependencies

We can see from our previous analysis of the ST-MR-IA-TD problem that none of the methods can provide any solution guarantees that are independent of v_D values. To show this effect in this simulation, we vary the maximum value for v_D^p in Γ^p for tasks. For tasks without dependencies, the maximum reward value is set to be 200. Figure 9 and Table 10 show the results as we gradually increase the maximum value for v_D^p from 400 to 4,000. While the average performance ratios remain high (with much larger standard deviations), the worst performance ratios for all four methods drop significantly as Fig. 9b shows, which complies with our theoretical results. However, we can see that *ResourceCentric* and *ResourceCentricApprox* perform notably better in terms of the worst performance ratios in this simulation, especially as the maximum value for v_D^p increases.

8.5 Time analysis

Finally, we provide time analysis for *AverageUtility*, *MaxUtility* and *ResourceCentric Approx* while gradually increasing the number of robots. The statistics are collected on

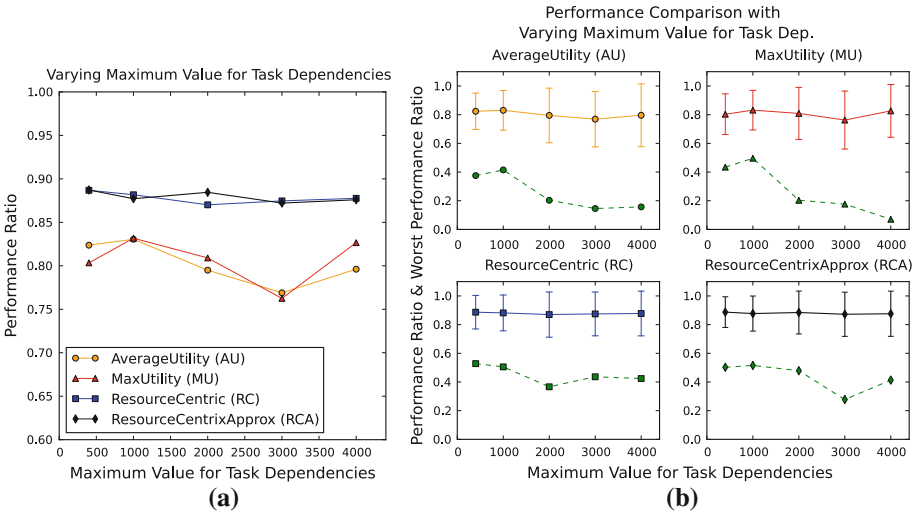

Fig. 9 Task allocation with task dependences with varying maximum value for v_D^p. **a** Average performance ratios. **b** Separate and more detailed average performance ratios with standard deviations. The *green* data points in each subgraph represent the worst performance ratios for that respective method (Color figure online)

Table 10 Outcome from *t*-tests for data points (each has 100 runs) in Fig. 9 (left to right) with $\alpha = 0.05$ (See text in Sect. 7.2 for explanation of the second row)

Paired-sample	RC, AU	RC, MU	RCA, AU	RCA, MU	RC, RCA	AU, MU
Sig. different?	yyyyy	yyyyy	yyyyy	yyyyy	nnnnn	nnnnn

our lab machines (2.67GHz) and the implementation is written in Java. As the time complexity of *ResourceCentric* is quadratic in $|C|$, we eliminate its performance from this first comparison for a clearer demonstration. While Fig. 10a shows the results for ST-MR-IA, Fig. 10b shows the results for ST-MR-IA-TD. Notice that the figures are scaled differently in this simulation. The running times of *ResourceCentricApprox* in both simulations are coarsely 100 times that of *AverageUtility* and *MaxUtility*. However, considering that $|R||T|$ is about 100 in these simulations, the results also comply with our theoretical analysis of the complexity for these methods. This suggests that *ResourceCentricApprox* indeed can be applied to problem instances of moderate sizes (e.g, 10–20 robots with 10–20 tasks to assign) within reasonable time limits (i.e., a few seconds), which is sufficient for most practical distributed robot systems.

A similar analysis is performed with *ResourceCentric* and *ResourceCentricApprox* to compare their performances. The results are shown in Fig. 11. One can see the effect that multiplying another $|C|$ has on the time performance.

8.6 Key findings from *coalition* results

In this section, we have provided simulation results for addressing the ST-MR-IA-TD problem. First of all, the results, along with our previous discussions, clearly demonstrate both theoretically and empirically that ST-MR-IA-TD is a more difficult problem. We have also confirmed the result on the hardness of approximating the ST-MR-IA-TD problem. Further-

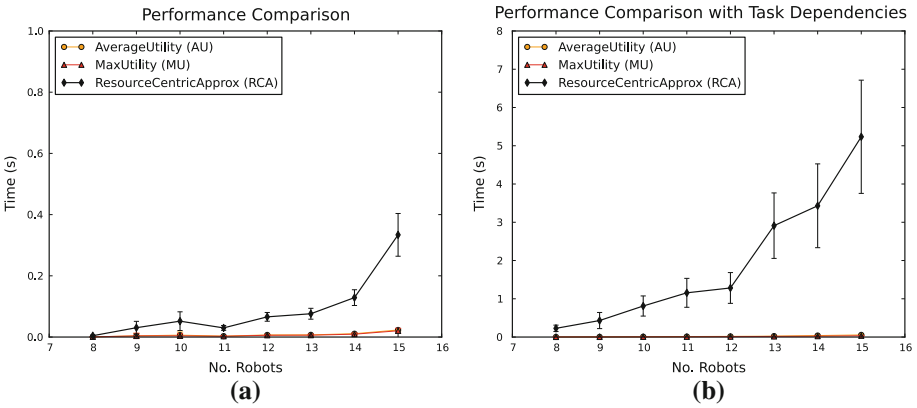

Fig. 10 Time analysis for all methods except *ResourceCentric*. **a** For ST-MR-IA. **b** For ST-MR-IA-TD (Note that different scales are used for (**a**) and (**b**))

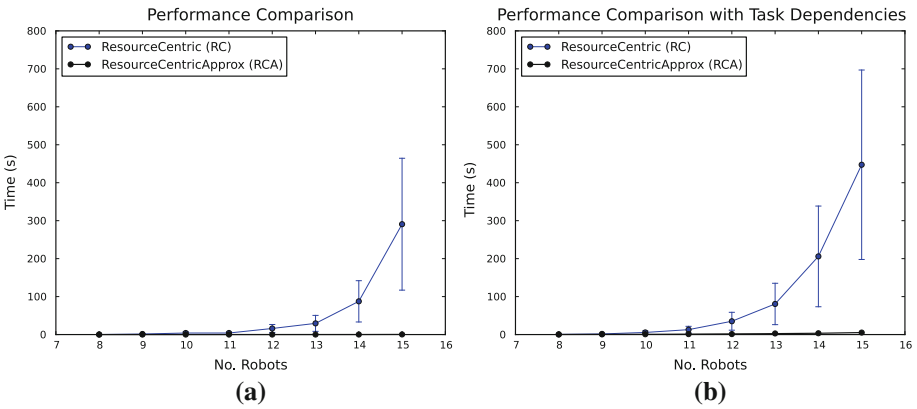

Fig. 11 Time analysis for *RC* and *RCA*. **a** For ST-MR-IA. **b** For ST-MR-IA-TD

more, we can see that in both problem formulations, *ResourceCentric* and *ResourceCentricApprox*, which consider inter-task resource constraints, perform better than *AverageUtility* and *MaxUtility* with significant differences. Thus, when one is working with relatively small problem instances (e.g., with <10 robots and <10 tasks), the *ResourceCentric* heuristic is recommended, due to its solution guarantees. On the other hand, for problem instances of moderate sizes (i.e., with 10–20 robots and 10–20 tasks), and when the time performance is more important, *ResourceCentricApprox* is recommended.

9 Conclusions and future work

In this paper, we first analyzed two natural heuristics for the ST-MR-IA problem. A new heuristic is then presented for the problem with solution guarantees. Results show that the solution quality of this heuristic is bounded by two factors – while one relates to a restricting parameter on the problem instance, the other is influenced by how assignments in the optimal solution interact with other assignments. Note that these two factors are not bounded by each

other, in the sense that while one can be greater than the other in one problem instance, it can be smaller in another. An algorithm is proposed to approximate this new heuristic for performance improvement. For more complicated scenarios, the ST-MR-IA problem is extended to incorporate general task dependencies. A result on the hardness of approximating this extended formulation of the problem is given. An algorithm that utilizes the methods for ST-MR-IA to address the extended problem is provided. Finally, simulation results are presented for both formulations, which show that these proposed methods do indeed improve performance.

In future work, we plan to facilitate our approach for even larger problem sizes (i.e., up to 50–100 robots). This requires us to further reduce the computational requirements. Another aspect is to efficiently implement the approach on distributed systems.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. 0812117. We gratefully acknowledge the valuable help of the anonymous reviewers, whose comments led to important improvements to this paper.

References

1. Abdallah, S. & Lesser, V. (2004). Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM international conference on intelligent agent technology* (pp. 162–168). China.
2. Atamturk, A., Nemhauser, G., & Savelsbergh, M. W. P. (1995). A combined Lagrangian, linear programming and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics, 1*, 247–259.
3. Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research, 4*(3), 233–235.
4. Dang, V. D., & Jennings, N. R. (2006). Coalition structure generation in task-based settings. In *Proceedings of the 17th European conference on artificial intelligence* (pp. 210–214).
5. Fanelli, L., Farinelli, A., Iocchi, L., Nardi, D. & Settembre, G. P. (2006). Ontology-based coalition formation in heterogeneous MRS. In *Proceedings of the 2006 international symposium on practical cognitive agents and robots* (pp. 105–116). New York: ACM.
6. Fua, C., & Ge, S. (2005). COBOS: Cooperative backoff adaptive scheme for multirobot task allocation. *IEEE Transactions on Robotics, 21*(6), 1168–1178.
7. Garey, M., & Johnson, D. (1978). "Strong" NP-completeness results: Motivation, examples, and implications. *Journal of ACM, 25*(3), 499–508.
8. Gerkey, B., & Mataric, M. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research, 23*(9), 939–954.
9. Hoffman, K., & Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science, 39*, 657–682.
10. Jones, E., Dias, M., & Stentz, A. (2011). Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous Robots, 30*, 41–56.
11. Lau, H. C., & Zhang, L. (2003). Task allocation via multi-agent coalition formation: taxonomy, algorithms and complexity. In *15th IEEE international conference on tools with artificial intelligence* (pp. 346–350).
12. Parker, L. E., & Tang, F. (2006). Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE, 94*(7), 1289–1305.
13. Rahwan, T., Ramchurn, S. D., Jennings, N. R., & Giovannucci, A. (2009). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research (JAIR), 34*, 521–567.
14. Sandholm, T., Larson, K., Andersson, M., Shehory, O., & Tohme, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence, 111*(1–2), 209–238.
15. Sandholm, T., Suri, S., Gilpin, A., & Levine, D. (2002) Winner determination in combinatorial auction generalizations. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems: Part 1* (pp. 69–76). New York: ACM.
16. Sariel, S. (2005). Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *Integrating planning into scheduling: Papers from the 2005 AAAI workshop* (pp. 27–33).

17. Service, T., & Adams, J. (2011). Coalition formation for task allocation: Theory and algorithms. *Autonomous Agents and Multi-Agent Systems, 22*, 225–248.
18. Shehory, O., & Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence, 101*(1–2), 165–200.
19. Tosic, P. T., & Agha, G. A. (2004). Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. In *Massively multi-agent systems* (pp. 104–120). Berlin: Springer.
20. Vig, L., & Adams, J. (2006). Multi-robot coalition formation. *IEEE Transactions on Robotics, 22*(4), 637–649.
21. Vig, L., & Adams, J. A. (2007). Coalition formation: From software agents to robots. *Journal of Intelligent and Robotic Systems, 50*(1):85–118.
22. Zlot, R. M. (2006). *An auction-based approach to complex task allocation for multirobot teams*. PhD thesis, Carnegie Mellon University, Pittsburgh, AAI3250901.
23. Zuckerman, D. (2007). Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing, 3*(1), 103–128.