Intelligent Service Robotics (2021) 14:501–517
https://doi.org/10.1007/s11370-021-00372-9

ORIGINAL RESEARCH PAPER

Scalable hedonic coalition formation for task allocation with heterogeneous robots

Emily Czarnecki[1] · Ayan Dutta[1]

Received: 20 September 2020 / Accepted: 2 June 2021 / Published online: 15 June 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract
Tasks in the real world are complex and often require multiple robots to collaborate to be serviced. In many cases, a task might require different sensory inputs and actuation outputs. However, allocating a large variety of sensors and/or actuators on a single robot is not a cost-effective solution—robots with different attributes must be considered. In this paper, we study coalition formation for such a set of heterogeneous robots to be allocated instantaneously to a set of tasks. Our proposed solution employs a *hedonic coalition formation* strategy based on a weighted bipartite matching algorithm. In our setting, a hedonic coalition game, a topic rooted in game theory, is used to form coalitions by minimizing the total cost of the formation and maximizing the overlap between required and allocated types of robots for each of the tasks. This approach guarantees a polynomial time complexity and Nash-stability. Numerical results show that our approach finds similar quality near-optimal solutions to existing approaches while significantly reducing the time to find them. Moreover, it easily scales to large numbers of robots and tasks in negligible time (1.57 sec. with 2000 robots and 400 tasks).

Keywords Task allocation · Coalition formation · Multi-robot systems

1 Introduction

To complete a given task with complex requirements we often see humans collaborate. Each contributor in the collaboration offers a different capability that aids in the task's successful completion. The completion of tasks in an autonomous environment implements a similar approach. In this environment, multiple robots each having their own set of capabilities—such as sensors or actuators—that collectively contribute to completing a given task. The work presented in this paper studies the coalition formation by heterogeneous robots, defined as the **S**ingle **T**ask, **M**ulti-**R**obot **I**nstantaneous task allocation problem (ST-MR-IA) [15,40,47] in the literature. Each robot has a different set of abilities in the form of sensors and/or actuators, and each task has a set of requirements in where a coalition of robots' collective abilities can complete a specific task. The objective of forming these coalitions is to minimize the *cost*, while the *value* of the formed coalitions is maximized. Finding the optimal solution for this problem is shown to be NP-Hard [40,41] although many approximation solutions with provable worst-case performance bounds exist [10,40,41].

Multi-robot coalition formation for task allocation has many potential real-world applications ranging from warehouse management to environmental monitoring. The objective is to partition the set of robots into non-overlapping subsets (coalitions), each of which will be allocated to a unique task and the member robots will then cooperate with each other to accomplish the allocated task. In this paper, we deal with the partitioning problem: Given a set of n robots and m tasks, how to form m coalitions such that some given criterion (e.g., the utility of the coalitions) is optimized. The required numbers and types of robots to complete a given task is assumed to be known *a priori*.

To solve this stated problem, we use a well-established game-theoretic concept, called hedonic coalition game [7]. The word 'hedonic' is originated from the Greek word ' hēdonikós' that means 'considered in terms of pleasure'. In a pure hedonic setting, each robot will consider joining a particular coalition if the experience of joining the coalition is 'pleasant', e.g., the robot's capabilities are maximally utilized and the cost of moving to the coalition is also low. To form a set of m non-overlapping hedonic coalitions, we have used a weighted bipartite graph matching technique

✉ Ayan Dutta
a.dutta@unf.edu

1 University of North Florida, Jacksonville, USA

along with an intelligent switch rule that makes the formed coalitions provably Nash-stable. Unlike previous approaches [40,47], which search through a large space to find a near-optimal solution, we jump-start with a 'good' solution and modify it as required while saving on both time and memory. Due to a generic graph formulation, our proposed solution can be easily extended to solve the **M**ulti **T**ask, **S**ingle **R**obot **I**nstantaneous task allocation problem (MT-SR-IA). To the best of our knowledge, this paper is the first to solve the coalition formation problem for task allocation with a group of heterogeneous robots using a hedonic coalition game formulation. Simulation results validate the theoretically proved stable nature of our solutions. The results show that our proposed algorithm can consistently yield near-optimal solutions (up to 94%).

Furthermore, we have compared our proposed approach with four existing approaches developed to provide an approximation solution to the heterogeneous coalition formation problem. Our approach is compared to two natural greedy heuristics: *MaxUtility* and *AverageUtility*. *MaxUtility* was first presented in [40] and further explored along with *AverageUtility* in [47]. To improve upon these two solutions, Zhang and Parker [47] proposed two new solutions to the ST-MR-IA problem: *ResourceCentric* and *ResourceCentricApprox*. We implemented similar solutions to the above-mentioned approaches, altered slightly, to adapt to the nature of the problem and models presented in this work. The comparison of the results revealed similar solutions in terms of the ratio to the optimal; however, our solution proved to be significantly faster when compared to all four algorithms. Out of these, *MaxUtility* had the fastest run times, but our solution was up to 41 times faster for the given experiment settings. The slowest was *ResourceCentric*. With a setting of $m = 4$ and $n = 12$, our solution was approximately 44, 500 times faster. Lastly, we implemented our proposed approach on a real TurtleBot 3 robot having a Raspberry Pi 3 computing board that is intended to reflect a more realistic environment in which the proposed solution would run on. It could find a solution for 100 robots and 10 tasks in 0.32 seconds.

A preliminary version of this work has recently appeared in the IEEE SMC conference [35]. We have made a major revision of that version and added the following: a more comprehensive literature review and an extensive complexity analysis. Moreover, we have compared our solution against four existing approaches, empirically validated the scalability of our approach, and thoroughly analyzed the effects of different parameters on our approach.

Our primary contributions in this article are listed as follows. To the best of our knowledge, this is the first work to use hedonic coalition formation for the multi-robot task allocation problem. Although the proposed algorithm is adopted from [6], we have modified it to fit the task allocation problem instead of the generic hedonic coalition formation approach taken in [6]. More specifically, we have

– used a weighted bipartite matching formulation instead of the unweighted matching shown in [6].
– theoretically analyzed the complexity and proved the Nash-stability of the proposed approach.
– proposed a novel utility function that is relevant to the heterogeneous task allocation problem studied in our paper.
– implemented and tested the proposed methodology to demonstrate the feasibility and scalability. Furthermore, we have compared our approach against four state-of-the-art techniques and shown that our proposed approach outperforms them in scalability and time while yielding similar quality solutions.

2 Background

A comprehensive taxonomy of multi-robot task allocation can be found in [15,19].

2.1 Task allocation

One of the earliest studies on coalition formation among agents for task allocation is presented by Shehory and Kraus [41]. Their work presented a greedy algorithm with a polynomial-complexity and it is guaranteed to find a solution within a factor of $(k+1)$ of the optimal solution, where k is the maximum size of any coalition formed. Recently, a correlation clustering technique is utilized to provide a near-optimal solution [13]. The solution applies linear programming to correlate robots to clusters of high similarity using cost and value functions. Dutta and Asaithambi [10] propose a variant of a classical weighted bipartite matching technique for the allocation of homogeneous robots. The proposed approach provides a similar near-optimality guarantee as [40,41] while incurring a linear time-complexity. This model is the primary motivation for using the weighted bipartite matching-based hedonic coalition formation solution proposed in this paper albeit for heterogeneous robots.

In real-world situations, the complexity of tasks may require multiple robots that offer different capabilities. With this, considerations have been given to the heterogeneous task allocation problem. The work presented in [40] provides two solutions for the coalition formation problem. One for a homogeneous and the other for a heterogeneous system. For the former, a dynamic programming-based algorithm is introduced, which can find an optimal solution in polynomial time. For the latter, the authors present an adjusted solution of the algorithm proposed in [41]. Their solution offers a polynomial solution with a complexity of

($O(n^{\frac{3}{2}}m)$), which improves upon Shehory and Kraus's solution with a complexity of $O(n^k m)$; exponential on the size of the largest coalition. However, both [40,41] report similar sub-optimality guarantees. In [47], the authors examine two natural greedy heuristics for the coalition formation problem within a heterogeneous environment and then present an improved heuristic in which inter-task resource constraints are taken into account as well as the expected loss of utility. Liemhetcharat and Veloso [23] proposed a coalition formation algorithm with heterogeneous agents where an agent learned about the capabilities of other agents over time and formed better coalitions. Here, a synergy graph is used to model how compatible any two agents are to form a coalition. The proposed solution takes a learning approach where the synergy graph developed is built using training observations. A similar synergy-graph model is used in [24] to study role assignments of ad-hoc agents in a coalition. In [45], the authors use such heterogeneous robots for a real-world application-box pushing. Tang and Parker [43] have proposed a distributed solution for such heterogeneous robots and applied it to a site cleaning scenario. Unlike ours, both task and coalition-related costs are considered. Rauniyar and Muhuri [34] pose the coalition formation problem with heterogeneous robots as a local search problem and apply a modified Genetic algorithm-based technique to find the best allocation. A Quantum Multi-Objective Particle Swarm Optimization approach for coalition formation in heterogeneous robot systems is studied in [28], where the authors could provide numerical results with up to 10, 000 robots in a natural disaster handling scenario. Similarly, the authors in [42] apply the task allocation problem to disaster rescue mission scenarios using a linear programming technique, where each task might have more than one objective. A related parallel multi-objective coalition formation approach is proposed in [2] and the authors test this technique within a CUDA programming framework. Similar to our approach in this paper, the authors in [44] propose a fully distributed technique where each robot decides to join a coalition based on its current members' available capabilities. Dutta et al. [11] proposed a distributed bipartite graph partitioning approach with region growing to provide near-optimal solutions for the ST-MR-IA task allocation problem. The authors divided the bipartite graph into k sub-graphs corresponding to the k agent types. The sub-graphs are then processed to allocate agents of the same type to tasks. Furthermore, the work provides an expansion to solve for the ST-MR-TD task problem and considers inter-task dependencies. In [3], the authors propose a distributed solution for dynamically arriving tasks where they might have various execution priorities. Nunes et al. [30] have recently proposed a taxonomy for such inter-related tasks, e.g., with temporal and ordering constraints. Luo et al. [25] have considered the heterogeneous task allocation problem for grouped tasks and present an auction-based solution to maximize payoff or minimize cost. Another auction-based strategy with using minimum spanning trees is presented in [21]. Recently, multiple auction-based methods for task allocation in a communication-limited environment is studied in [32]. Afghah et al. [1] have studied a similar task allocation problem using a leader-follower scheme, where a leader robot (for each task) recruits followers for task completion. A swarm intelligence approach for task allocation where the robots can accomplish multiple tasks in the RoboCup Rescue challenge is proposed in [8]. In [18], the authors present a communication-free task allocation scheme for a robot swarm, where the result converges to a system-level Bayesian Nash equilibria. Sarkar et al. [39] considers the capacity-constrained vehicle routing problem as the base model for task allocation and provides a scalable heuristic handling up to 400 robots and 2000 tasks in simulations.

Additional complexities to the heterogeneous task allocation problem such as uncertain costs have also been taken into account. Uncertainty is captured in the edge costs of transport graphs by Prorok [33]. The author has used redundant robots to combat the uncertainty of travel times to reduce the waiting times at the desired goal location. They present a polynomial-time algorithm using distributive aggregate functions to assign redundant robots to paths to minimize average waiting time. In [16], the authors propose an auction-based solution for applications where the robots might gain or lose some of their available capabilities, whereas in our paper, the capabilities remain constant. The limited capabilities of the robots are considered in [46], where the objective was to maximize the number of serviced tasks by the robots, whereas we consider a situation where all the tasks need to be allocated. Nam and Shell [29] consider uncertain costs and risk levels associated with unknown settings such as surface topology—one solution allows the input of risk tolerance preference affecting the optimal assignment and in the second approach, the risk position does not affect the optimal assignment.

2.2 Hedonic games

The concept of hedonic games, originally presented by Dreze and Greenberg [9], found its first applications in economics. Specifically, the idea that individuals form teams to accomplish activities and the motivation for an individual to join a team is based on preferences. Since then several studies have been conducted to further define and understand hedonic games and its applications. Bogomolnaia and Jackson's work [7] is a primary example of one of the earlier studies researching the stability of coalition structures in hedonic games. In this work, the authors review the settings for hedonic and non-hedonic situations and the existence of stability in hedonic settings. Further studies have continued to research stability and complexity that can be found in [4,14]. In the former, the authors research computational complexity to achieve stable results. They consider hedonic games with additively separable utilities and different stable outcomes such as Nash and individual stability. The latter study considers a player's preference for one coalition over another. This work offers solutions considering preference restrictions and guarantees the existence of stable outcomes.

A type of hedonic games known as *Fractional Hedonic Games* has also amassed a decent amount of attention. Aziz et al. [5] focused on the applications and stability of fractional hedonic games. In this variant, the utility is an average value a player credits the others in the coalition. The authors have presented an algorithm to compute a core-stable outcome for such a game. The work presented in [6] also considers fractional hedonic games. The authors analyze various graph topologies in order to model these games and their complexities when aiming to achieve a Nash-stable outcome.

Game theory has previously been used for robot planning problems [22]. However, specifically proposing hedonic games for multi-robot planning appears to have received less attention. Research works in the last decade by Saad et al. [36–38] focused on using hedonic games for self-forming coalitions in different multi-agent systems including unmanned air vehicles, wireless communication agents, and cognitive radio networks. The solutions aim to reduce the number of preference relations built in the process [36–38]. On the other hand, we study a similar problem but for heterogeneous robots. Recently, Jang et al. [17] have used a hedonic game-based formulation for multi-robot task allocation albeit for a homogeneous system.

3 Model and notations

Let $R = \{r_1, r_2, \cdots, r_n\}$ denote a set of n robots. Each robot r_i has the following attributes: position P_i and a set of capabilities (e.g., sensors, actuators) $S_{r_i} = \{s_1, s_2, \ldots s_k\}$. W.l.o.g. We make the following assumptions: 1) a robot's capabilities $|S_{r_i}| \geq 1$; 2) a robot cannot have duplicate capabilities (e.g., multiple GPS's); and 3) each robot is localized in the environment using an on-board GPS or an overhead camera. A set of m $(< n)$ tasks are available in the environment denoted by $T = \{t_1, t_2, \ldots t_m\}$. A task t_j has the following attributes: position P_j and capability requirements $S_{t_j} = \{s_1, s_2, \ldots s_l\}$. Each task's requirements will adhere to $|S_{t_j}| \geq 1$ as well and a task can require multiple robots with the same capability to be completed. Note that $\bigcup_m S_{t_j} \subseteq S$ and $\bigcup_n S_{r_i} = S$, where S denotes the set of all possible capabilities.

A coalition $c \subseteq R$ is a group of robots assigned to complete a task. A coalition structure CS is a set of disjoint coalitions, which cover all the robots. Let $CS = \{c_1, c_2, \ldots c_m\}$ denote a set of non-overlapping coalitions where each coalition is matched with a single task and as such $|CS| = |T| = m$ and $\cup_{j=1}^{m} c_j = R$.

Value Function. This function determines the non-negative value a robot offers a task, i.e., how effective that robot is when contributing to the needs of any particular task. In our scenario, the value of a robot allocated to a task is defined by how many capabilities the robot r_i can contribute to the task t_j's requirements. On the other hand, an unused capability is a wasted resource negatively affecting the overall value a robot offers that task. The better value a robot can offer a task, the more likely the robot will choose that task to be assigned to. The value function is a weighted formula of capability matching and is defined as the following:

$$val(r_i, t_j) = \begin{cases} bI - pO, & \text{if } bI - pO > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where b is the benefit ascribed to I, which is the size of the common subset of capabilities belonging to r_i and t_j's requirement sets, i.e., $I = |S_{r_i} \cap S_{t_j}|$. p is the penalty received for each capability not utilized, and O is a subset of capabilities belonging to r_i not needed by the task, i.e., $O = |S_{r_i} \setminus (S_{r_i} \cap S_{t_j})|$. If the result is negative, the value is set to 0 indicating this robot offers no value to the task being considered. It is important for the value to be non-negative in order to achieve a Nash-stable outcome [6]. Both b and p are constants, values of which can be set depending on how much desired benefit or penalty can be ascribed to the matching. For the purpose of this paper, $b = 1$ and $p = 0.1$. The benefit and penalty weights work in delicate balance with each other and can offer flexibility in the importance of matched and unused capabilities.

The penalty p should be chosen to avoid a negative $val(\cdot, \cdot)$. A penalty may only be added if at least one capability match is made; otherwise, the value defaults to zero as defined in Eq. 1. Therefore, the maximum value O can have is the $\max(|S_{r_i}|) - 1, \exists r_i \in R$, which we denote by $max(O)$. As the robots do not have duplicate capabilities (e.g., two LIDARs), $max(|S_{r_i}|)$ is equivalent to the number of capability types. For example, with maximum of three capabilities, $max(O) = 3 - 1$. To ensure $pO \leq bI$, the penalty value must be: $p \leq \frac{bI}{max(O)}$ where $I \geq 1$. On the other hand, with $b = 1$ and $max(O) = 2$, the upper bound on p is 0.5. It is important to note that While there is a maximum bound on the penalty p, the value used for p is not required to be the maximum in all the cases.

The used value for p depends on the specifics of the application. The penalty is used to aid in reducing wasted resources. If wasted resources cannot be tolerated, the penalty should be set to its maximum. In our problem where we consider three types of capabilities, the maximum penalty of 0.5

gives it a significant weight and thus brings the overall offered value down. This greatly restricts the robots from potentially relocating to a better coalition, which was the primary motivation for setting $p = 0.1$.

Cost Function For a robot to service an assigned task, the robot must travel to the task location to complete it. When a robot considers coalitions to join, it must consider the cost to travel to its chosen task as well. Travel not only impacts battery life but also overall wear and tear of the robot. Therefore, it is an important factor to consider in the formulation. In our formulation, the cost is determined by the distance traveled. With self-forming coalitions, the cost function will aid the robots in deciding if they want to stay in their current coalitions, or if other coalitions are preferred based on how far they have to travel to reach the tasks. To make sure that the cost function is always within [0, 1], we first normalize the raw distance between a task and a robot. Next, to guarantee that it produces a positive number when combined with $val(\cdot,\cdot)$, we take the **c**omplement of the normalized **c**ost, i.e., subtract it from 1, called $cc()$, which is formally defined as:

$$cc(r_i, t_j) = 1 - \left(\frac{d(r_i, t_j)}{\sqrt{length^2 + width^2 + 1}}\right) \tag{2}$$

where d denotes the Euclidean distance between a robot and a task and $length$ and $width$ are the dimensions of the environment.

Utility Function The utility function indicates overall how well a robot fits within its coalition in terms of value and cost. When a robot considers a coalition, it will use the (non-negative) utility metric to determine how well it is suited to service the task. The higher the utility, the better suited the robot is to the task being considered. It is defined as:

$$u(r_i, t_j) = val(r_i, t_j) + cc(r_i, t_j) \tag{3}$$

To follow the hedonic coalition game model, each robot's utility must solely depend on the members of the coalition [7]. To adhere to this property a robot r_i's value can *only* be determined based on the other robots in the coalition under consideration. r_i must take into account the other robots' capabilities contributed to the coalition already as it indicates the capabilities still required by that task.

Problem Objective The objective is to have self-forming set of coalitions, where the final coalition structure maximizes the utility. The utility of the coalition structure is the sum of utilities of all the robots allocated to coalitions and is defined as follows:

$$u(CS) = \sum_{j=1}^{m} \sum_{i=1}^{|c_j|} u(r_i, t_j) \tag{4}$$

The objective of the coalition formation problem for multi-robot task allocation is to find a coalition structure CS^* that offers the maximum utility and it can be formally represented as follows:

$$CS^* = arg \max_{CS \in \zeta} u(CS)$$

where ζ denotes the set of all possible coalition structures. After an initial allocation, each robot will consider other coalitions in the CS and relocate to a task, if necessary, in order to increase its utility. This will result in a final Nash-stable coalition structure.

4 Hedonic coalition formation game

In a hedonic setting, originally used to model social/economic scenarios, the agents (robots in our case) have a preference as to which coalition they wish to belong to [9]. Some examples of such situations are social groups, (soccer) teams, organizations, etc. An agent's motivation to belong to a coalition can be quantified by its utility, which is *only* dependent on the other members within that coalition [7]. One such example is a political party, where a member's utility depends on the party's values and identities of other members [6]. We can formalize the hedonic coalition formation by defining the following key properties [7,37]:

1. The utility of a robot depends solely on the other robots in the coalition that it currently belongs to.
2. The coalitions are formed based on the preferences the robots have over the set of all possible coalitions.

The concept of Nash stability states that a coalition structure CS is stable if there exists no robot that can improve its utility by moving to another coalition in CS. Simply put, all robots are 'happy' with their current utilities and are in the coalitions they most prefer. Our objective in this work is to follow the properties of the hedonic model to build self-forming coalitions, which result in a final coalition structure that is Nash-stable. The Nash-stability is formally defined below [37].

Definition 1 A coalition structure CS is *Nash-stable* if $\forall r_i \in R$, $c_{r_i} \succeq_{r_i} c \cup \{r_i\}$ for all $c \in CS \cup \{\emptyset\}$.

where c_{r_i} is the coalition r_i currently belongs to and $\succeq_{r_i}$ denotes r_i's preference for c_{r_i} over c. In layman's terms, every robot r_i will prefer to belong to its current coalition than joining a different coalition c. To form such coalitions, we utilize concepts presented in [6,10] focusing on solutions based on bipartite matching. To model the coalitions, we use an undirected, weighted bipartite graph unlike the unweighted graph model used in [6]. We first perform a matching on a bipartite graph modeled with the robots and the tasks. Thereafter, we give the robots the opportunity to improve their utilities by relocating to different coalitions as discussed next.

4.1 Maximum bipartite graph matching

We begin with a maximum weighted bipartite matching formulation to establish initial coalitions where exactly one robot is assigned to each task. Although we have used the classical weighted bipartite matching algorithm (Algorithm 1) presented in [26], any other weighted matching algorithm can be used in its place. We define our undirected, weighted bipartite graph as $G = (\{V, U\}, E, W)$. V and U are two sets of vertices where V represents the robots and U represents the tasks. E is the set of all possible edges between $u - v$ pairs where $u \in U$, $v \in V$. W is the set of edge weights. The weight of an edge is the utility between a robot-task pair and it is calculated using Eq. 3.

Definition 2 (Matching) A subset of edges $M \subseteq E$ is called a matching if the edges do not share any end vertices.

Algorithm 1 will produce such an edge set, called *matching*.

Definition 3 (Maximum Weighted Matching) A matching M is called a maximum weighted matching if the sum of the edge weights of M is the highest.

Algorithm 1 aims to find such a matching. This ensures that the robot-task pairs present in the matching yield a high utility.

Definition 4 (Maximum Matching) A matching M is the maximum if the maximum possible vertices from G are covered in M.

Definition 5 (Perfect Matching) A matching M is perfect if all the vertices in G are covered by the matching.

We are interested in finding a maximum matching using Algorithm 1, which will ensure that our initial coalition structure will consist of all mutually best robot-task pairs. This consequently will help us to find a high-quality allocation (i.e., coalition structure) quite quickly. However, as in our setting $n > m$, we will not be able to find a perfect matching. Next, we briefly explain the working principle of Algorithm 1; more details can be found in [26]. Although we have used a centralized version of the approximation matching algorithm presented by Manne and Bisseling in [26], a parallel version presented in the same paper can also be employed for speed-ups.

Algorithm 1 will take each task and match it to a robot where the robot-task matching is *mutually best* for each other. A mutually best robot-task pair means the edge $\{u, v\}$ is the maximum weighted edge among all edges incident to u and v. First, for each $v \in V$ we obtain the potential matches and store in S_v (lines $1 - 2$). We initialize S_v to U because initially, every $u \in U$ is a potential match. We initialize M to an empty set. While there are tasks remaining to be matched to a robot, for each such task, we find a mutually best robot pairing (lines $8 - 9$). $bm(\cdot)$ function finds the best match, i.e., the highest utility pair, of any node. Once a robot-task pair is found to be mutually best for each other (i.e., $bm(u) = v$ and $bm(v) = u$), they are added to M. This matched pair is also added to the set T_A so that it can be used later for assigning other tasks. After matching the initial set of mutually best robot-task pairs, the remaining unmatched tasks are assigned in a similar fashion (lines $11 - 17$). This process terminates when each task is assigned exactly one robot to it as $m < n$. However, not all the robots might be assigned at this stage.

Algorithm 1: Weighted Maximum Bipartite Matching

```
Input: $G(\{V, U\}, E, W)$: A weighted bipartite graph
where $V$ = set of robots and $U$ = set of tasks
Output: $M$: a weighted maximum matching.
1  for each $v \in V$ do
2      $S_v \leftarrow U$;
3  $M \leftarrow \emptyset$;
4  $T_A \leftarrow \emptyset$;
5  for each $u \in U$ do
6      $v \leftarrow bm(u)$;
7      if $bm(u) = v$ then
8          /*u and v are mutually best for each other*/
9          $M \leftarrow M \cup \{u, v\}$;
10         $T_A \leftarrow T_A \cup \{u, v\}$;
11 while $T_A \neq \emptyset$ do
12     $u \leftarrow$ end vertex $\in U$ from an edge in $T_A$;
13     $T_A = T_A \setminus \{u, v'\}$ where $\{u, v'\} \in M$;
14     for each $u \in S_v$ where $u$ is unmatched do
15         if $bm(bm(u)) = u$ then
16             $T_A = T_A \cup \{u, bm(u)\}$;
17             $M = M \cup \{u, bm(u)\}$;
18 return $M$;
```

4.2 Hedonic coalition formation algorithm

After the initial maximum weighted matching is found, it is utilized for the final hedonic coalition formation where a robot may choose to leave its current coalition for another if doing so will increase its utility. We first create the initial coalition structure, i.e., where each task has exactly one robot assigned to it. Next, the robots with improving deviations consider moving to other coalitions when there is an opportunity to do so.

Improving Deviation A robot r_i has an improving deviation if it can improve its utility by leaving its current coalition c_{r_i} and moving to another coalition c [6]. Any robot with an improving deviation will belong to a set defined as $R_{ID} = \{r_1, r_2, \ldots r_N\}$ where $R_{ID} \subseteq R$.

To develop self-forming coalitions, a robot r_i with an improving deviation will choose a coalition to relocate to based on its *preference*. Therefore, for a robot considering relocation, we define a preference relation between itself and all other coalitions. We define the preference relation as follows:

$$c_1 \succeq_{r_i} c_2 \leftrightarrow u(r_i, c_1) \geq u(r_i, c_2) \quad (5)$$

We slightly abuse the notation here by using c_1 and c_2 instead of t_1 and t_2 in the utility function w.l.o.g. In layman's terms, a robot r_i will strictly or equally prefer coalition c_1 over c_2 *iff* r_i's utility associated to c_1 is greater than or equal to r_i's utility in c_2. The preference relation is specific to some robot r_i. In our approach, only one robot may change coalitions at a time. This robot will have the lowest edge weight, i.e., the lowest utility among the robots in R_{ID}. The robot in question will define a preference relation for itself by considering all possible coalitions in the current CS.

Moving to a New Coalition After the robot with the lowest edge weight in R_{ID} generates a preference relation between itself and the current coalitions in the CS, it will join another coalition that increases its utility. A robot r_i will leave its coalition by using the following a *switch rule*, similar to that of [37]: Given a coalition structure $CS = \{c_1, c_2, \ldots c_m\}$, the candidate robot $r_i \in R_{ID}$ will leave its current coalition c_{r_i} and join another coalition c, *iff* $c \cup \{r_i\} \succ_{r_i} c_{r_i}$. We extend the unweighted hedonic coalition formation algorithm in [6] to a weighted setting and form hedonic coalitions.

Algorithm 2: Hedonic Coalition Formation

```
   Input: $R$: A set of Robots; $T$: A set of Tasks
   Output: $CS$: A Final Coalition Structure
1  Create the bipartite graph $G(\{V, U\}, E, W)$;
2  $M \leftarrow$ Weighted maximum matching from Algorithm 1;
3  $CS \leftarrow \emptyset$;
4  $Covered \leftarrow$ Set of all robots part of $M$;
5  for each $v \notin Covered$ do
6      choose $u \in U \mid \{u, v\} \in E$ and $w_{u,v} \geq w_{u',v}\ \forall\, u' \neq u$;
7      $c_u \leftarrow c_u \cup v$;
8      $Covered \leftarrow Covered \cup v$;
9  Update $CS$;
10 Compute the set of robots with Improving Deviation, $R_{ID}$;
11 while $R_{ID} \neq \emptyset$ do
12     Select $r_i \in R_{ID}$ with lowest edge weight;
13     $r_i$ considers switch operations using the preference relation
       calculated in Eq. 5;
14     $r_i$ selects a coalition using the switch rule;
15     Update $CS$;
16 return $R_{ID}, CS$;
```

Algorithm 2 defines the pseudo-code for coalition formation using principles in fractional hedonic games. Given an undirected and weighted bipartite graph, the algorithm will return a Nash-stable coalition structure CS. It will begin by first computing a weighted maximum matching defined in Algorithm 1. The matching returned by Algorithm 1 establishes the initial set of m coalitions (line 4). Next, for each robot (v) not yet in a coalition, it will select a task using a greedy methodology. A robot will allocate itself to a task by selecting the edge between itself and some task (u) such that the edge has the maximum weight for that robot among all potential tasks. This will continue until all robots have a task allocation (lines $5-8$). This provides an initial CS where each robot is allocated to some task. Finally, the algorithm selects the robot in R_{ID}, the set of robots with improving deviations, with the minimum edge weight and allows this robot to move to another coalition using the switch rule. The robot will leave its current coalition and choose the coalition that maximizes its utility, i.e., the task that has the highest edge weight for that specific robot. The current coalition structure CS is thereby updated. When the set R_{ID} is empty, i.e., no robot can improve its utility by changing coalitions, CS is considered stable and the final coalition structure CS is returned.

A Nash stable coalition structure is not necessarily the optimal coalition structure. Instead, it is a notion of equilibrium—each robot is 'happy' in its current coalition and does not feel the need to switch to a different coalition to improve its utility. Once the agreement is reached among the robots, they will not need to coordinate anymore, and thus the requirements of inter-agent communication and computation are not required. This results in communication and computation time reduction, which bears a high amount of significance, especially in a distributed decision-making setting [17]. On the other hand, Nash stability by default implies individual stability where no robot can move to a different coalition without making its members earn less utility [4]. After the final CS is formed, the robots will move to their allocated tasks. Inter-robot collisions can be avoided while minimally increasing the initially estimated path lengths by using techniques proposed in [12].

Lemma 1 *There always exists a Nash-stable solution for the proposed model.*

Proof The proof of this directly follows from Observation 3 in [6], which finds that any hedonic game played on a graph G with non-negative edge weights will find a Nash-stable coalition structure. As our graph structure also has non-negative edge weights (i.e., utilities), the proof follows. □

The existence of a Nash-stable solution can also be argued (informally) in the following way. As we keep track of coalition changes by the robots and corresponding change in any

Table 1 Summary of time complexity comparison

Algorithm	Reference	Complexity
Average Utility	[40,41,47]	$\mathcal{O}(\min(m,n)\cdot m\mathcal{C})$
Max Utility	[40,41,47]	$\mathcal{O}(\min(m,n)\cdot m\mathcal{C})$
ResourceCentric	[47]	$\mathcal{O}(\min(m,n)\cdot m^2\mathcal{C}^2)$
ResourceCentricApprox	[47]	$\mathcal{O}(\min(m,n)\cdot m^2n\mathcal{C})$
Our approach	this paper	$\mathcal{O}(mn^2)$

robot's potential utility gain possible by such a robot, following [27], it can be categorized as a potential function (lines $11-15$ in Algo. 2). Any such game is called a potential game and Monderer and Shapley [27], has proved that it would converge to a Nash-stable solution.

Theorem 1 *The final coalition structure is Nash-stable.*

Proof We prove this by contradiction. Let us assume that all the robots in R except r_∞ are stable, i.e., r_∞ is changing its coalition infinite times resulting in a non-Nash stable coalition structure. r_∞ prefers a coalition c_j over its current coalition c_i. According to the switch rule, c_j is the most preferred coalition of r_∞. Following our utility function, once a new robot joins a new coalition c_j, the utility of c_j for the non-member robots either remains the same (r_∞'s capabilities are mutually exclusive) or goes down (the capabilities overlap). Therefore, once r_∞ moves to c_j, there will not be a better coalition c_k that r_∞ can possibly move to next and thus will become stable. Combining this contradiction with the existence of a Nash stable coalition structure (Lemma 1) proves the theorem. □

Discussion on Complexity Time complexity of Algorithm 1 is $\mathcal{O}(mn)$ [26]. If an optimal algorithm for maximum bipartite matching such as Hungarian [20] is used instead, the complexity would become $\mathcal{O}((m+n)^3)$.

The time complexity of Algorithm 2 would primarily depend on lines 2, $5-8$, and $10-15$. As discussed above, line 2 will incur a complexity of $\mathcal{O}(mn)$. Lines $5-8$ will incur a time complexity of $\mathcal{O}(n)$ as all the robots are unassigned at this point in the worst case scenario. The complexity of line 10 in Algorithm 2 is $\mathcal{O}(mn)$ as each robot will check whether it can improve its utility by moving to any of the existing coalitions, the maximum count of which is m. As the R_{ID} set can maximally have n robots in it and each robot $r_i \in R_{ID}$ will be checking for the *switch*, lines $11-15$ will incur a complexity of $\mathcal{O}(mn)$. Thus, the worst-case time complexity of Algorithm 2 is $\mathcal{O}(mn^2)$.

Now, we compare the time complexity of our approach with that of the four existing state-of-the-art algorithms in Table 1. $\mathcal{C}$ denotes the total number of coalitions possible with n robots ($=2^n$). Therefore, it should be noted that all these algorithms will incur a high, exponential time complexity whereas our proposed hedonic game-based coalition formation approach incurs only a polynomial time complexity. This helps our algorithm to be highly scalable whereas the compared state-of-the-art algorithms cannot handle more than only tens of robots. One should also note that the *ResourceCentric* and *ResourceCentricApprox* algorithms [47] incur high space complexities since they store all the coalitions in the memory. As the number of coalitions ($\mathcal{C}$) grows exponentially with n, they will incur exponential space complexities. On the other hand, our proposed method does not take this approach and requires only polynomial space ($\mathcal{O}(mn)$). These practical implications are also demonstrated in the next section.

5 Experiments

5.1 Settings

We have implemented our proposed hedonic coalition formation algorithm using the Java programming language. The tests were run on a laptop with an Intel $i7-3615$QM processor and 16GB RAM. The number of robots (n) has been varied between $[4, 2000]$, and the number of tasks (m) has been varied between $[2, 400]$. We have made sure that in no test case the number of tasks exceeds 50% of the number of robots used. The distinct 2D locations of the robots and the tasks are randomly generated from a bounded square area with sides of length 100m. The total number of possible capabilities ($|S|$) was set to 3 and the capability distribution was randomly generated for both robots and tasks. To begin, robots received a random combination between one and three unique capabilities. To determine how many requirements each task would receive, the sum of capabilities among the robots was randomly partitioned based on the number of tasks in the environment. The sum of each type of capability given to the robots was used to randomly generate the capability requirements of the tasks. Experiments were run with a baseline benefit (b) of 1.0 and penalty (p) of 0.1 unless otherwise mentioned. Each setting was run 20 times with an average result calculated and illustrated in the next section. The bars in the plots indicate the maximum and the minimum value obtained for any particular metric.

We have compared the performance of our algorithm against four previous approaches. The first two approaches—*MaxUtility* and *AverageUtility*—implemented for comparison are greedy algorithms developed in [41] as well as modifications of this approach in [40] and [47]. In both algorithms—beginning with the first task—all coalitions of a size equal to that of the robot requirement for the given task are generated. Then each candidate coalition is evaluated. With *MaxUtility*, the total utility of the coalition is considered. The alternate approach—*AverageUtility*—considers the mean utility of the candidate coalition. Given that the goal is to maximize utility, the candidate coalition with the highest (*MaxUtility*), or the highest average (*AverageUtility*) utility is greedily assigned to the task being considered. These task and robots are then removed from consideration, and the process repeats with the remaining tasks and robots.

The other two solutions implemented for comparison are developed in [47]. The *ResourceCentric* and *ResourceCentricApprox* solutions are greedy heuristics that consider inter-task resource constraints. For both, the first step is to generate all possible coalitions constrained to the maximum size. In *ResourceCentric*, an undirected graph is generated where each node is a coalition, and edges are added where a conflict exists between two coalitions. A conflict occurs when two coalitions contain the same task or robots as a task cannot be assigned to more than one set of robots and a robot cannot be part of multiple coalitions. After the generation, while the graph is not empty, for each coalition (node) and for each of its neighbors, a value ρ is calculated, which is the utility of the coalition minus the sum of the conflicting assignments multiplied by the utility of the neighbor coalition. The coalition that maximizes ρ is selected and this coalition, its neighbors, and connecting neighbors are removed from the graph. The key differences in *ResourceCentricApprox* as compared to *ResourceCentric* is an approximation of the calculation ρ and the use of hash tables to track conflicts instead of an undirected graph.

5.2 Results

A number of experiments were run to determine the overall quality and effectiveness of the solutions. These experiments and their results are presented next.

5.2.1 Utility comparisons

First, we are interested in the investigation of the approximation ratio (higher is better, 1 being the optimal solution) achieved by our proposed method to an optimal solution. To this end, we have implemented a brute-force method [31] to obtain the optimal solution and we compute the ratio of our solution's utility to that of the optimal solution. The brute-force method employs the same value, cost, and utility calculations presented Eqs. 1, 2, and 4, respectively. The results are presented in Fig. 1. The plots demonstrate the near-optimal nature of our solution.

For example, with $m = 2$, the highest and the lowest approximation ratios obtained by our approach are 0.91 and 0.78 for $n = 4$ and 10, respectively. The approximation ratios were quite similar with $m = 4$ as compared with $m = 2$, and we obtained an approximation ratio of 0.84 with $n = 8$ and 0.79 with $n = 12$. Next we look at how well our solution performs in terms of ratio to the optimal in comparison to the four additional greedy solutions implemented.

The result in comparing our approach to the *MaxUtility* solution is shown in Fig. 1a. Here we can observe that *MaxUtility* provided better solutions with $m = 2$; however, the solution degraded with $m = 4$ and our solution provided better approximation results. The highest approximation ratio for *MaxUtility* was 0.96 with $m = 2$ and $n = 12$, and the lowest was 0.70 with $m = 4$ and $n = 10$. In Fig. 1b, we see the results of our solution compared with *AverageUtility*, which shows *AverageUtility* provided a lower quality solution in all settings than our approach. The highest approximation ratio for *AverageUtility* was 0.86 with $m = 2$ and $n = 4$, and the lowest was 0.53 with $m = 4$ and $n = 12$. In Fig. 1c, we present the comparison results against *ResourceCentric* approach. Result illustrates that *ResourceCentric* provides a better solution in some cases with $m = 2$, but with $m = 4$, our approach provides a better solution. The highest approximation ratio for *ResourceCentric* was 0.89 with $m = 2$ and $n = 10$, and the lowest was 0.66 with $m = 4$ and $n = 10$. In Fig. 1d, we compare *ResourceCentricApprox* against our algorithm. *ResourceCentricApprox*, similarly to some of the earlier comparisons, resulted in better solutions at $m = 2$. However, did not perform as well as approach for $m = 4$. The highest approximation ratio for *ResourceCentricApprox* was 0.94 with $m = 2$ and $n = 6$, and the lowest was 0.70 with $m = 4$ and $n = 10$.

Following [10,40], we are also interested to investigate how many times out of the 20 simulation runs, we get a *good* solution, i.e., at least 80% of the optimal. The result is presented in Fig. 2. We observe that for both $m = 2$ and 4, in about half of the test runs, our proposed algorithm achieves a solution that is within 80% of the optimal (Fig. 2a).

In comparing our approach with the results achieved with *MaxUtility* as shown in Fig. 2b, we see that *MaxUtility* outperforms our approach with $m = 2$ but does not perform as well with $m = 4$. *MaxUtility* can achieve a solution that is within 80% of the optimal 90% of the time, and at minimum 50% of the time within 90% of the optimal for $m = 2$. For $m = 4$, in most cases *MaxUtility* achieves solutions within 80% or 90% of the optimal less than 30% of the time. *AverageUtility* overall was completely outperformed by our approach. The result is presented in Fig. 2c. In all scenarios save one, *AverageUtility* could not achieve solutions within 80% or 90% of the optimal in half of the simulation runs. The *ResourceCentric* heuristic performed well with $m = 2$ as can be seen in Fig. 2d. It provided a good solution–within 80% of the optimal–typically above 80% of the time. In at least half of the runs, it was able to produce a solution within 90% of the optimal. As we have seen with the other comparison algorithms though, it does not perform as well as our approach with $m = 4$ and does not achieve a solution within either 80% or 90% of the optimal more than 30% of the time.

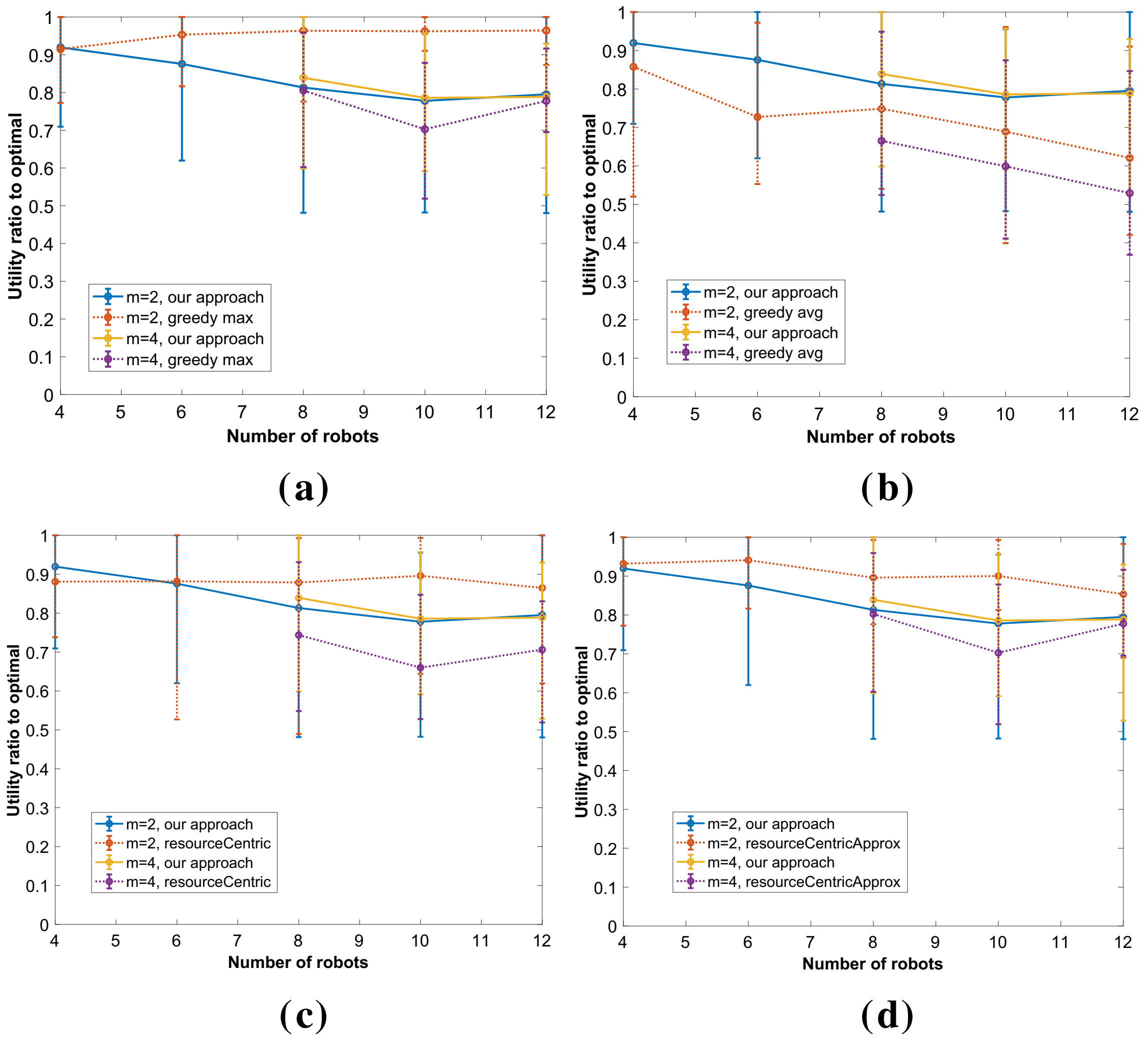


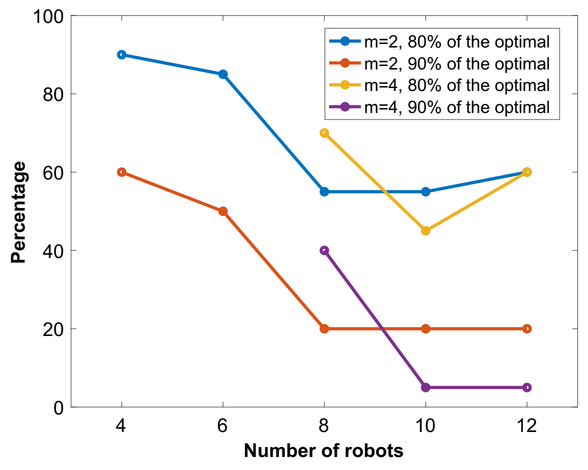
Fig. 1 Approximation ratio to the optimal: **a** *MaxUtility*, **b** *AverageUtility*, **c** *ResourceCentric*, and **d** *ResourceCentricApprox*

A similar assessment can be made for the *ResourceCentricApprox* approach presented in Fig. 2e. It provides a good solution in most cases for $m = 2$ at least 80% of the time. It performs more comparably with our approach with $m = 2$ when seeking a solution within 90% of the optimal. Again, similar to the other algorithms used for comparison, it does not perform as well when $m = 4$ and in all cases except where $n = 8$ does not achieve a solution within either 80% or 90% of the optimal more than 30% of the time. Overall we can conclude that in some cases the comparison algorithms provide better solutions when looking at ratio to the optimal solution, but in all cases, our solution shows better results as the number of tasks increases.

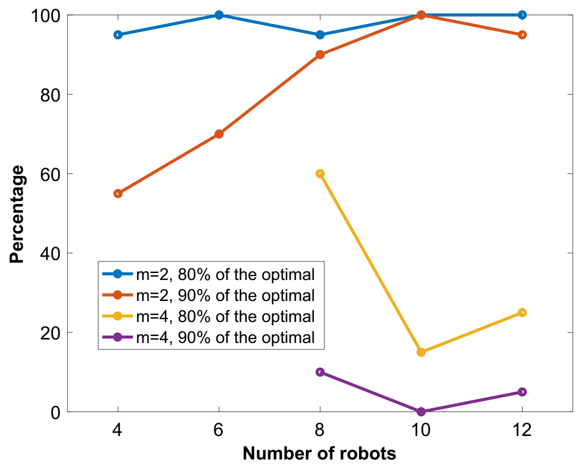
5.2.2 Impact of benefit and penalty

Next we investigate how the benefit (b) and penalty (p) weights affect the solution in terms of utility and proximity to the optimal solution. In these experiments, we chose baseline values of $b = 1.0$ and $p = 0.1$. To understand the impact of these two values, we ran an experiment with b values varying between 1.0 and 1.75 with increments of 0.25, and an additional experiment with p values between $[0.1, 0.4]$ with increments of 0.1. First, we review the impact of the b values on the approximation ratios (Fig. 2f). Note that, in this case, p was static at 0.1. Overall, we can observe that the change in b does indeed impact the approximation ratio. Furthermore, we can observe that the baseline benefit of 1.0 produces the lowest approximation ratios. Overall, when looking at all scenarios, a benefit of $b = 1.50$ offers, in general, higher approximation ratios. At $m = 4$ and $n = 12$ the solution produces a mean approximation ratio of 0.85. The highest mean approximation ratio of 0.94 was achieved with $b = 1.75$ at $m = 2$ and $m = 4$.

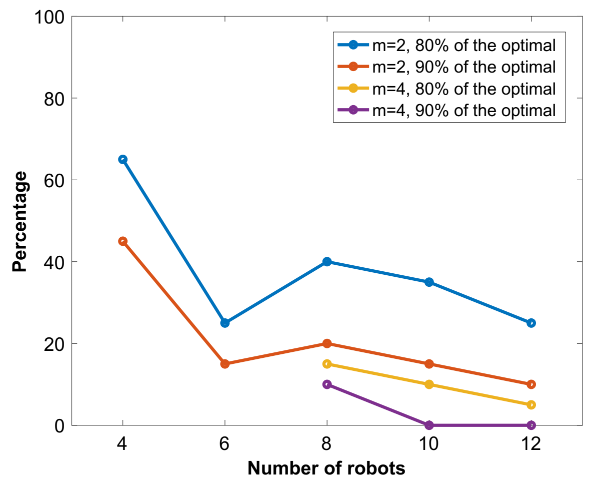
We now analyze the results with varying penalty values. Our results presented in Fig. 3a show the approximations



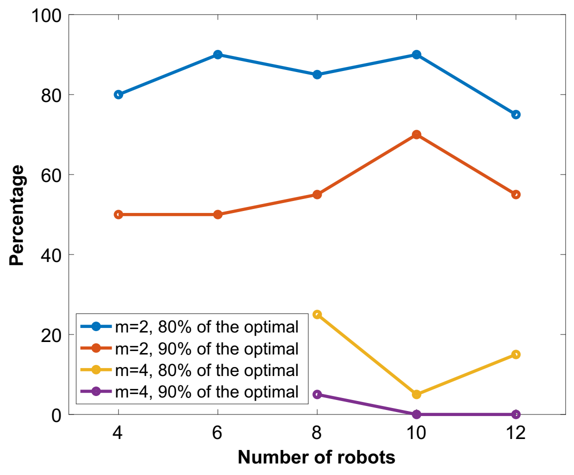
(a)



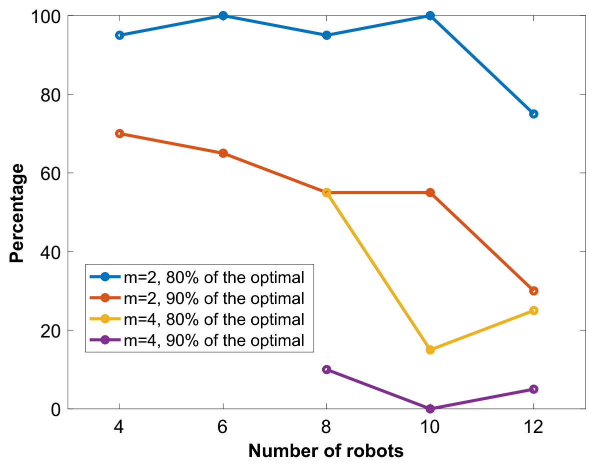
(b)



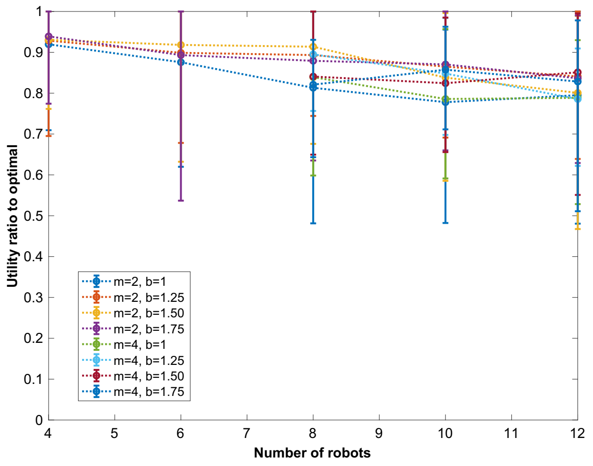
(c)



(d)

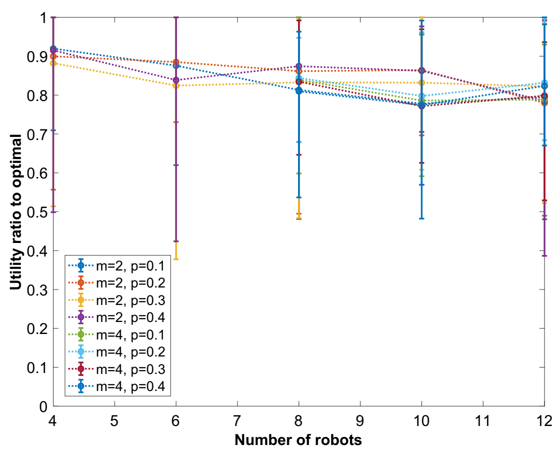


(e)

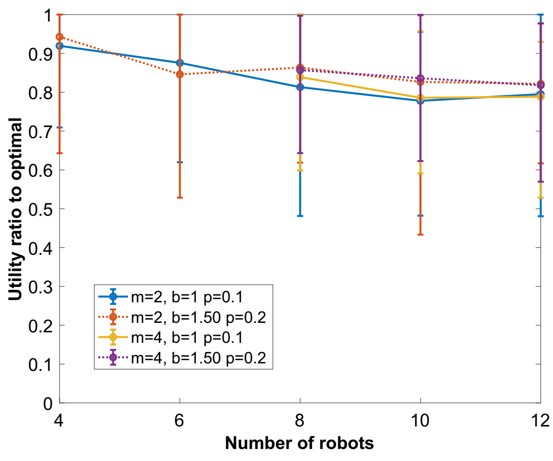


(f)

Fig. 2 Percentage of tests in which the algorithm generated a coalition structure with utilities 80% and 90% of the optimal: **a** Our, **b** *MaxUtility*, **c** *AverageUtility*, **d** *ResourceCentric*, and **e** *ResourceCentricApprox*. **f** Approximation ratio to optimal with varying benefit (*b*) values



(a)



(b)

Fig. 3 **a** Approximation ratio to optimal with varying penalty (p) values; **b** approximation ratio to optimal with benefit = 1.5 and penalty = 0.2

ratios with varying p values and static b value of 1. As with benefit, we observe that the penalty value does impact ratio to the optimal. However, $p = 0.1$ did produce the highest singular mean approximation ratio of 0.92 at $m = 2$ and $n = 4$. When looking at the quality of solutions in all scenarios though, a penalty of $p = 0.2$ results in a higher approximation ratios producing a ratio of 0.83 with $m = 4$ and $n = 12$. Finally, after identifying benefit and penalty values that perform well, we were interested in reviewing the results in terms of approximation ratio when employing those values. We can observe these results in Fig. 3b. In this result, we compare the baseline benefit and penalty (1.0, 0.1, respectively), to our observed better performing benefit and penalty (1.50, 0.2, respectively). As can be seen, overall higher approximation ratios were achieved with the latter. This is particularly notable with $m = 4$ and n values of 8, 10, 12. The highest approximation ratio of 0.94 was achieved at $m = 2$ and $n = 4$. Approximation ratios of 0.86, 0.84, 0.82 were achieved with $m = 4$ and $n = 8, 10, 12$, respectively.

5.2.3 Performance comparisons

Next, we are interested in understanding how well our algorithm performs in terms of time in comparison with the four comparison algorithms. Additionally, we are interested in how our solution scales with a large set of robots. Run time comparison result against the *MaxUtility* solution is shown in Fig. 4a). As can be observed, only in the first scenario of $m = 2$ and $n = 4$, *MaxUtility* outperforms our solution with respect to run time. At very low numbers of tasks and robots, *MaxUtility* has very few possible coalitions to consider, and given the simplicity of the greedy choices in this algorithm, it makes sense that this algorithm would be quick initially. However, it quickly grows in run time as the number of robots and tasks increases due to the exponential nature of possible coalitions to consider. With $m = 4$ and $n = 12$, *MaxUtility* takes 0.933ms, as compared to our 0.031ms, resulting in our approach being approximately 30 times faster.

Similar results can be observed with *AverageUtility* as illustrated in Fig. 4b. Again, the first scenario of $m = 2$ and $n = 4$ shows a faster run time than our solution, but its run time increases significantly as the number of robots and tasks increases. With $m = 4$ and $n = 12$, *AverageUtility* takes 1.27ms, as compared to our 0.031ms, resulting in our approach being approximately 41 times faster. Comparison result against the *ResourceCentric* algorithm is shown in Fig. 4c. Here we can see our solution significantly outperforms *ResourceCentric*. As with *MaxUtility* and *AverageUtility*, the algorithm considers all possible coalitions of a maximum size, but performs more complex calculations taking resource constraints into account, resulting in a slower performance. At $m = 4$ and $n = 12$, *ResourceCentric* takes 1336.38ms with our solution taking 0.031ms, making our solution approximately 44, 500 times faster. The *ResourceCentricApprox* algorithm is modification on *ResourceCentric* to improve its performance with run time. This can be observed in Fig. 4d. We can see that it does indeed perform much better in terms of time as compared with *ResourceCentric*; however, it still does not perform as well as our solution. At $m = 4$ and $n = 12$, *ResourceCentricApprox* takes 2.81ms. As compared with our solution with a run time 0.031ms.—making our solution approximately 90 times faster. When tested against these algorithms with a relatively larger robot set, similar trends are noticed. n is varied between {15, 20} and m is fixed to 6 for this test. The results are presented in Fig. 5. Our presented approach earns higher utilities for both values of n while taking the lowest time to find a solution among the tested algorithms.

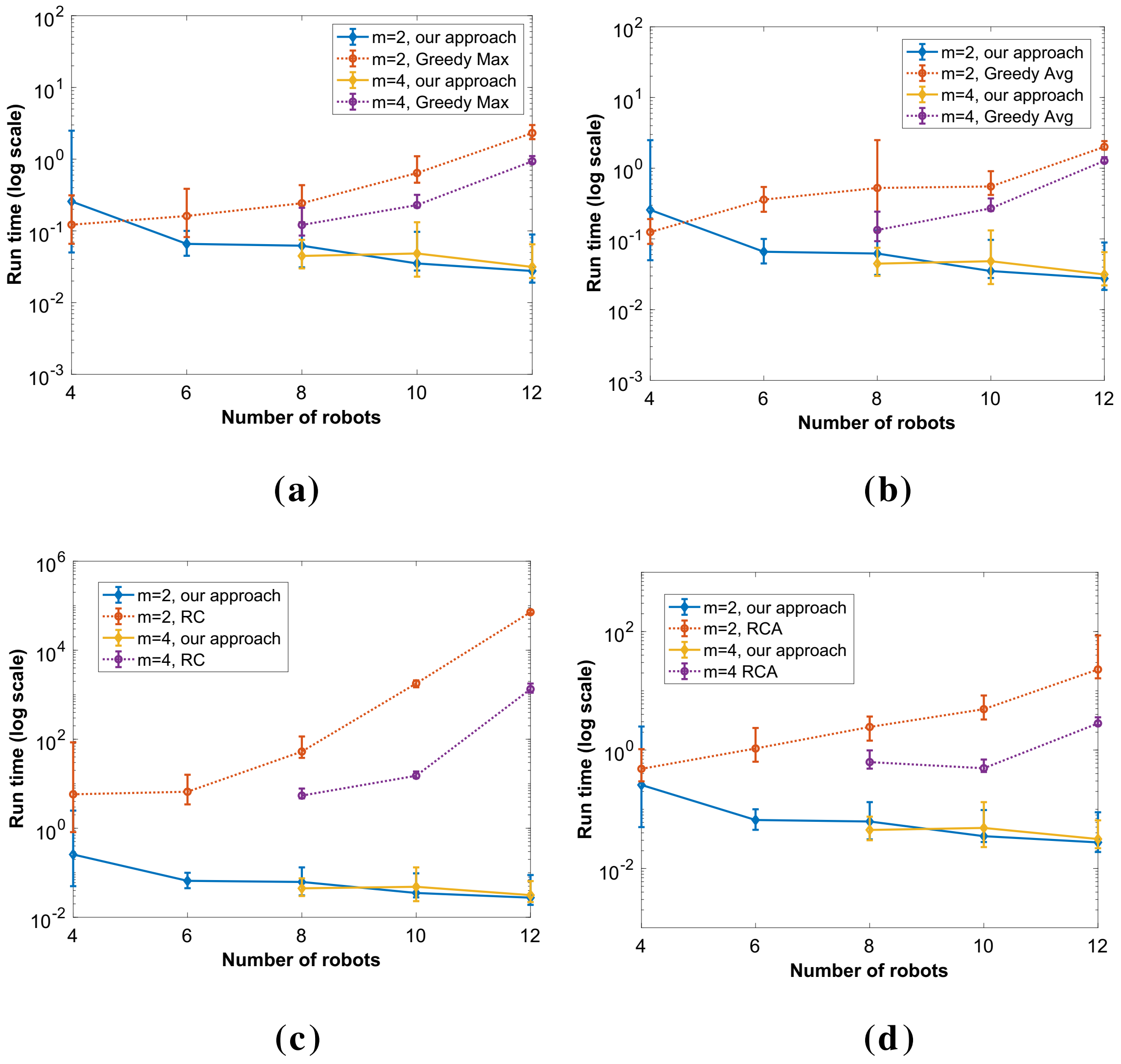


Fig. 4 Run time comparison to: **a** *MaxUtility*, **b** *AverageUtility*, **c** *ResourceCentric*, and **d** *ResourceCentricApprox*

The *ResourceCentric* algorithm performed the worst in terms of run time—the maximum being 301.53 sec. for $n = 20$ whereas our algorithm took 0.003 sec.

As stated earlier, our goal in this study is to develop a scalable algorithm that produces near-optimal solutions. Figure 6 demonstrates the scalability of our proposed solution for a moderately large multi-robot system—the maximum run time is found to be 0.40ms. and the mean run time was 0.19 ms. with $n = 100$ and $m = 10$. A negligible number especially considering the fact that for this particular setting, the astronomical number of possible coalition structures is 2.75×10^{93}.

Lastly, we investigate distances the robots travel to get to their assigned tasks. The robots want to minimize the traveled distance amount to move to a specific task. Therefore, distance is an important performance metric. In Fig. 6b, we see that the total distance traveled by the robots increases linearly with more robots irrespective of the task counts. As with less number of tasks in the environment, each robot needs to travel more distance to reach a particular task because of the uniform distribution of tasks and robots, it is worth noting that with less tasks, the robots travel more. This is consistent with the results found in [10]. If we observe the total distance traveled by the robots in Fig. 6b, we can see the maximum distance traveled was with 2 tasks and 100 robots. The mean distance traveled was 3839.64 meters and the maximum was 5673.29 meters. With 10 tasks and 100 robots, the mean distance traveled was 2572.36 meters and a maximum of 3573.26 meters.

5.2.4 Switch rule analysis

The number of coalition switches does not show a clear trend (Fig. 7a). However, we notice that the average number of

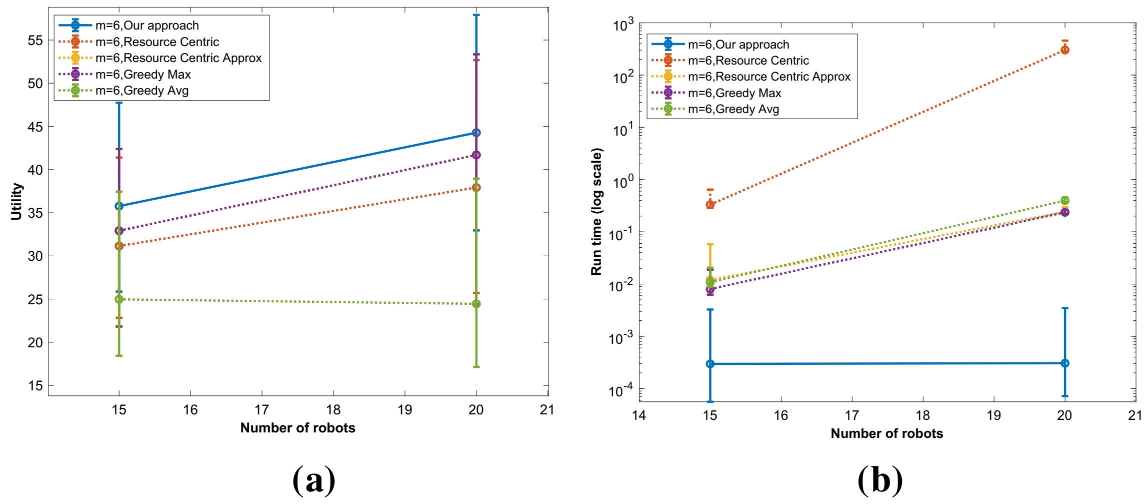


Fig. 5 Comparison of **a** utility and **b** run time metrics against *MaxUtility*, *AverageUtility*, *ResourceCentric*, and *ResourceCentricApprox*

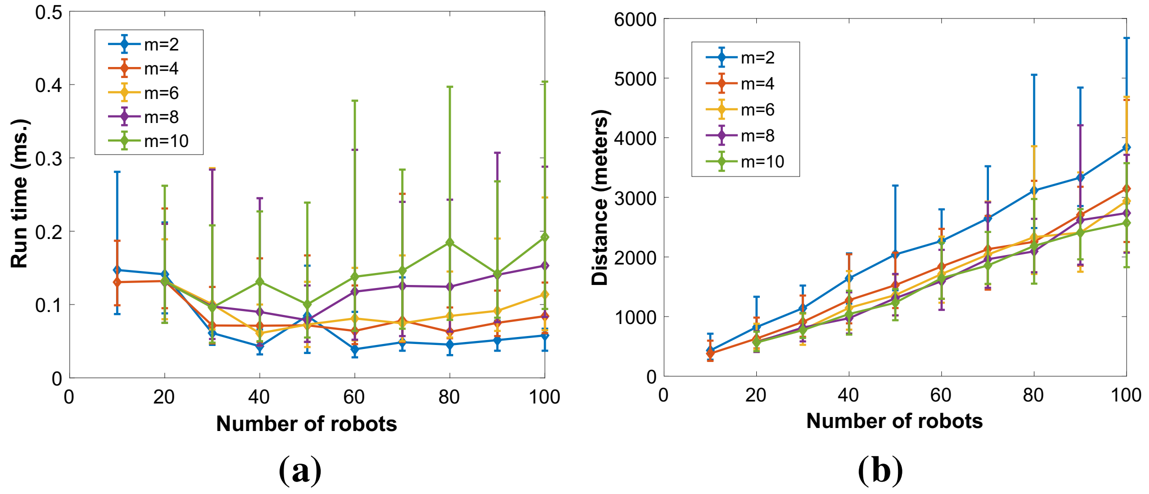


Fig. 6 **a** Run time of the proposed approach, **b** total distance traveled by the robots to reach the allocated tasks

switches occurred across the tested m and n values is almost negligible and generally higher number of tasks correspond to higher number of switches. As the robots have more coalitions to relocate to, the number of switches is more probable with more tasks. The small, finite number of switches also demonstrates the stability of the solution. For 2 tasks, 100 robots the mean number of switches was 0, for 4 tasks, 100 robots the mean switches was 0.2, for 6 tasks and 100 robots mean switches was 0.55, for 8 tasks 100 robots mean switches was 1 and for 10 tasks 100 robots mean switches was 1 as well. Looking at the maximum number of switches that took place for the $m = 2, 4, 6, 8, 10$, the maximum switch counts were 1, 2, 3, 5, and 6, respectively, showing again an increase in switches as the number of tasks increases. Finally, we want to demonstrate the usefulness of the switch rule used in our model. We see in Fig. 7b that generally with a higher number of coalition switches the robots were able to increase the total utility, the maximum being 24% with six switches. Next, we show how with increasing switches (only for three switches), the solution quality improves (Fig. 7c). We observe that on average, the initial *CS* utility is 89.33% of the final utility, with a steady increase as more switches are performed—demonstrating the anytime nature of the switch rule.

To showcase that the proposed algorithm captures the task requirements, we present a case study with $n = 6$ and $m = 2$, where there are two types of robots: three iRobot Roombas and two iRobot Bravas (each having a singular capability—vacuuming and moping, respectively). The tasks, t_1 and t_2, are to vacuum and mop two rooms located at (10, 10) and (20, 20). As the first room is significantly bigger than the second, t_1 needs three Roombas and one Brava, whereas t_2 needs one Roomba and one Brava robot for cleaning. The robots are located in a room at the location (15, 15).

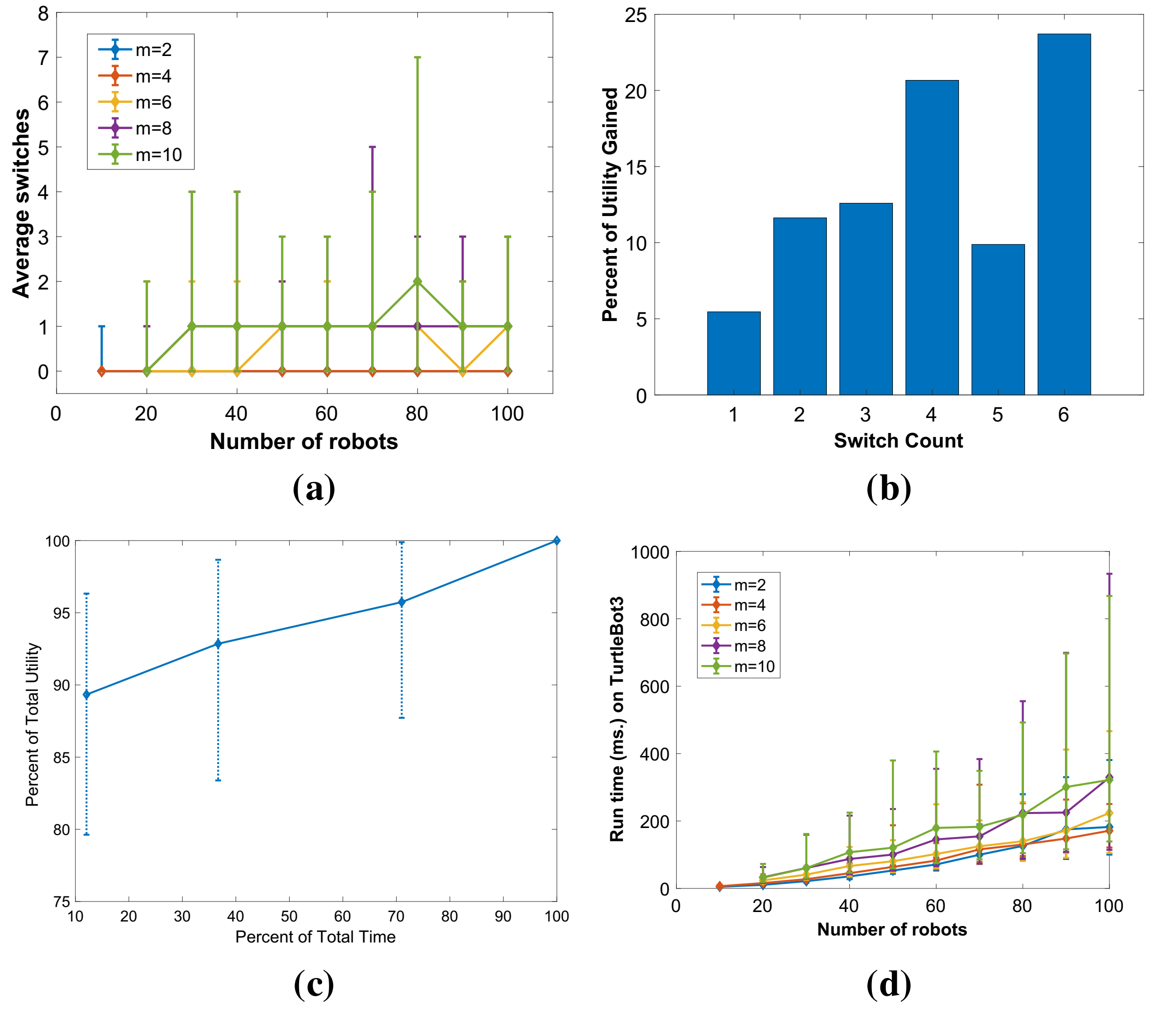


Fig. 7 **a** Average number of switches performed, **b** utility gained (%) for different number of switches, **c** change in utility over time ($n = 100$, $m = 10$), d) run time of our proposed solution on a TurtleBot 3

Using our algorithm, t_2 gets assigned a single Roomba and t_1 gets assigned four Roombas and two Bravas.

5.2.5 Implementation on a TurtleBot 3

We are also interested in investigating the feasibility of running our algorithm on a system where computational capabilities are limited. For this, we implemented the algorithm on a TurtleBot 3 robot equipped with a Raspberry Pi 3. We used the same experiment settings when testing the scalability varying tasks from [2, 10] and robots from [10, 100] taking the average of 20 runs. With $m = 10$ and $n = 100$ we achieved a mean run time of 0.32 seconds and a maximum run time of 0.86 seconds (Fig. 7d). This demonstrates the scalability of the proposed algorithm on a real hardware platform with limited resources.

5.2.6 Scalability tests

Finally, we empirically analyze the scalability of our proposed approach for large-scale multi-robot systems. We use the same combined numbers of tasks and robots as used in [39]. The results are shown in Fig. 8. Similar to our earlier tests with up to 100 robots, we see that the distances traveled by the robots to reach the allocated task locations increase linearly with n. The run time with 2400 robots and tasks is a negligible 1.57 sec. For the same number of robots and tasks, the proposed approach in [39] took about 50 sec. Due to the hardware differences, we do not, however, compare these results quantitatively.

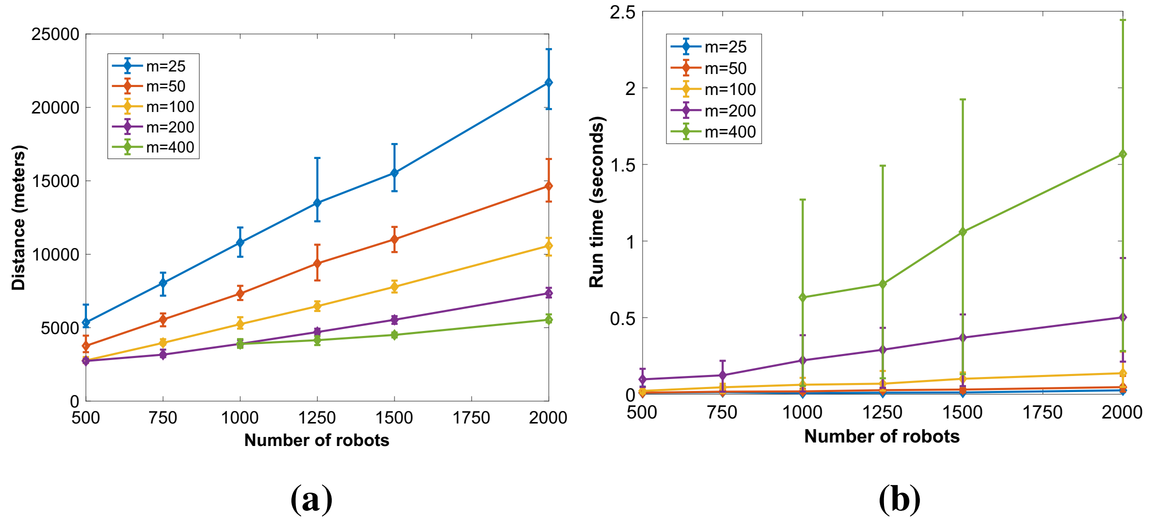


Fig. 8 Scalability testing: **a** cost of the solution; **b** run time of the algorithm

6 Conclusions and future work

The research presented in this work proposes a novel approach to the heterogeneous multi-robot task allocation problem. To the best of our knowledge, this approach is the first to solve the coalition formation problem for task allocation with a group of heterogeneous robots using a hedonic coalition game formulation. This problem has numerous real-world applications; however, finding the optimal solution for the studied problem is shown to be NP-hard in the literature. Our solution offers an approach utilizing hedonic coalition and concepts from graph matching. Results show that our proposed solution is fast, does produce near-optimal solutions, and can be applied for a large multi-robot system as well as offering the ability to calibrate the outcome with the benefit and penalty weights. Additionally, simulation on a TurtleBot 3 shows that our solution can be utilized in practical application settings where computing capabilities may be limited. Comparisons against four state-of-the-art approaches resulted in comparable outcomes in terms of utility ratio; however, our solution provided a significant improvement in run time. The increasing use of robots and robot teams across industries lends itself to revealing new ways in which robots can coordinate and fulfill task requirements. Among some of the possible expansions of this work are developing a distributed approach in order to avoid one point of failure, considering inter-task dependencies, and incorporating uncertainty into the model.

References

1. Afghah F, Zaeri-Amirani M, Razi A, Chakareski J, Bentley E (2018) A coalition formation approach to coordinated task allocation in heterogeneous uav networks. In: 2018 Annual American Control Conference (ACC), pp. 5968–5975. IEEE
2. Agarwal M, Agrawal N, Sharma S, Vig L, Kumar N (2015) Parallel multi-objective multi-robot coalition formation. Expert Syst Appl 42(21):7797–7811
3. Ayari E, Hadouaj S, Ghedira K (2017) A dynamic decentralised coalition formation approach for task allocation under tasks priority constraints. In: 2017 18th International Conference on Advanced Robotics (ICAR), pp. 250–255. IEEE
4. Aziz H, Brandl F (2012) Existence of stability in hedonic coalition formation games. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 763–770. International Foundation for Autonomous Agents and Multiagent Systems
5. Aziz H, Brandt F, Harrenstein P (2014) Fractional hedonic games. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, pp. 5–12. International Foundation for Autonomous Agents and Multiagent Systems
6. Bilo V, Fanelli A, Flammini M, Monaco G, Moscardelli L (2018) Nash stable outcomes in fractional hedonic games: existence, efficiency and computation. J Artif Intell Res 62:315–371
7. Bogomolnaia A, Jackson MO (2002) The stability of hedonic coalition structures. Games Econ Behav 38(2):201–230
8. Dos Santos F, Bazzan AL (2011) Towards efficient multiagent task allocation in the robocup rescue: a biologically-inspired approach. Auton Agents Multi-Agent Syst 22(3):465–486
9. Dreze JH, Greenberg J (1980) Hedonic coalitions: optimality and stability. Econometrica 48(4):987
10. Dutta A, Asaithambi A (2019) One-to-many bipartite matching based coalition formation for multi-robot task allocation. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 2181–2187. IEEE
11. Dutta A, Czarnecki E, Asaithambi A, Ufimtsev V (2019) Distributed coalition formation with heterogeneous agents for task allocation. In: The 32nd International Flairs Conference, pp. 116–119. AAAI Press
12. Dutta A, Dasgupta P (2017) Bipartite graph matching-based coordination mechanism for multi-robot path planning under communication constraints. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 857–862. IEEE

13. Dutta A, Ufimtsev V, Asaithambi A, Czarnecki E (2019) Coalition formation for multi-robot task allocation via correlation clustering. Cybern Syst 50(8):711–728
14. Gairing M, Savani R (2010) Computing stable outcomes in hedonic games. In: International Symposium on Algorithmic Game Theory, pp. 174–185. Springer
15. Gerkey BP, Matarić MJ (2004) A formal analysis and taxonomy of task allocation in multi-robot systems. Int J Robot Res 23(9):939–954
16. Irfan M, Farooq A (2016) Auction-based task allocation scheme for dynamic coalition formations in limited robotic swarms with heterogeneous capabilities. In: 2016 International Conference on Intelligent Systems Engineering (ICISE), pp. 210–215. IEEE
17. Jang I, Shin HS, Tsourdos A (2018) Anonymous hedonic game for task allocation in a large-scale multiple agent system. IEEE Trans Robot 34(6):1534–1548
18. Kanakia A, Touri B, Correll N (2016) Modeling multi-robot task allocation with limited information as global game. Swarm Intell 10(2):147–160
19. Korsah GA, Stentz A, Dias MB (2013) A comprehensive taxonomy for multi-robot task allocation. Int J Robot Res 32(12):1495–1512
20. Kuhn HW (1955) The hungarian method for the assignment problem. Naval Res Logist Q 2(1–2):83–97
21. Lagoudakis MG, Berhault M, Koenig S, Keskinocak P, Kleywegt AJ (2004) Simple auctions with performance guarantees for multi-robot task allocation. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 1, pp. 698–705. IEEE
22. LaValle SM, Hutchinson S (1993) Game theory as a unifying structure for a variety of robot tasks. In: Proceedings of 8th IEEE International Symposium on Intelligent Control, pp. 429–434. IEEE
23. Liemhetcharat S, Veloso M (2012) Modeling and learning synergy for team formation with heterogeneous agents. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, pp. 365–374. International Foundation for Autonomous Agents and Multiagent Systems
24. Liemhetcharat S, Veloso M (2012) Weighted synergy graphs for role assignment in ad hoc heterogeneous robot teams. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5247–5254. IEEE
25. Luo L, Chakraborty N, Sycara K (2014) Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks. IEEE Trans Robot 31(1):19–30
26. Manne F, Bisseling RH (2007) A parallel approximation algorithm for the weighted maximum matching problem. In: International Conference on Parallel Processing and Applied Mathematics, pp. 708–717. Springer
27. Monderer D, Shapley LS (1996) Potential games. Games Econ Behav 14(1):124–143
28. Mouradian C, Sahoo J, Glitho RH, Morrow MJ, Polakos PA (2017) A coalition formation algorithm for multi-robot task allocation in large-scale natural disasters. In: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1909–1914. IEEE
29. Nam C, Shell DA (2017) Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation. IEEE Robot Autom Lett 2(1):193–200
30. Nunes E, Manner M, Mitiche H, Gini M (2017) A taxonomy for task allocation problems with temporal and ordering constraints. Robot Auton Syst 90:55–70
31. Orlov M (2002) Efficient generation of set partitions. Engineering and Computer Sciences, University of Ulm, Tech. Rep, Germany
32. Otte M, Kuhlman MJ, Sofge D (2020) Auctions for multi-robot task allocation in communication limited environments. Auton Robots 44(3):547–584
33. Prorok A (2019) Redundant robot assignment on graphs with uncertain edge costs In Distributed Autonomous Robotic Systems. Springer, Berlin
34. Rauniyar A, Muhuri PK (2016) Multi-robot coalition formation problem: Task allocation with adaptive immigrants based genetic algorithms. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 000137–000142. IEEE
35. Czatnecki E, Dutta A (2019) Hedonic coalition formation for task allocation with heterogeneous robots. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 1024–1029. IEEE
36. Saad W, Han Z, Basar T, Debbah M, Hjorungnes A (2009) A selfish approach to coalition formation among unmanned air vehicles in wireless networks. In: 2009 International Conference on Game Theory for Networks, pp. 259–267. IEEE
37. Saad W, Han Z, Basar T, Debbah M, Hjorungnes A (2011) Hedonic coalition formation for distributed task allocation among wireless agents. IEEE Trans Mobile Comput 10(9):1327–1344
38. Saad W, Han Z, Basar T, Hjorungnes A, Song JB (2010) Hedonic coalition formation games for secondary base station cooperation in cognitive radio networks. In: 2010 IEEE Wireless Communication and Networking Conference, pp. 1–6. IEEE
39. Sarkar C, Paul HS, Pal A (2018) A scalable multi-robot task allocation algorithm. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–9. IEEE
40. Adams JA, Service TC (2011) Coalition formation for task allocation: theory and algorithms. Auton Agents Multi-Agent Syst 22(2):225–248
41. Shehory O, Kraus S (1998) Methods for task allocation via agent coalition formation. Artif Intell 101(1–2):165–200
42. Su X, Wang Y, Jia X, Guo L, Ding Z (2018) Two innovative coalition formation models for dynamic task allocation in disaster rescues. J Syst Sci Syst Eng 27(2):215–230
43. Tang F, Parker LE (2007) A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation. In: ICRA, pp. 3351–3358
44. Tošić PT, Agha GA (2014) Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. In: International Workshop on Massively Multiagent Systems, pp. 104–120. Springer
45. Vig L, Adams JA (2006) Multi-robot coalition formation. IEEE Trans Robot 22(4):637–649
46. Wu D, Zeng G, Meng L, Zhou W, Li L (2017) Gini coefficient-based task allocation for multi-robot systems with limited energy resources. IEEE/CAA J Automatica Sinica 5(1):155–168
47. Zhang Y, Parker LE (2013) Considering inter-task resource constraints in task allocation. Auton Agents Multi-Agent Syst 26(3):389–419

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.