Auton Agent Multi-Agent Syst (2011) 22:225–248
DOI 10.1007/s10458-010-9123-8

# Coalition formation for task allocation: theory and algorithms

**Travis C. Service · Julie A. Adams**

Published online: 24 February 2010


**Abstract** This paper focuses on coalition formation for task allocation in both multi-agent and multi-robot domains. Two different problem formalizations are considered, one for multi-agent domains where agent resources are transferable and one for multi-robot domains. We demonstrate complexity theoretic differences between both models and show that, under both, the coalition formation problem, with $m$ tasks, is NP-hard to both solve exactly and to approximate within a factor of $O(m^{1-\epsilon})$ for all $\epsilon > 0$. Two natural restrictions of the coalition formation problem are considered. In the first situation agents are drawn from a set of $j$ types. Agents of each type are indistinguishable from one another. For this situation a dynamic programming based approach is presented, which, for fixed $j$ finds the optimal coalition structure in polynomial time and is applicable in both multi-agent and multi-robot domains. We then consider situations where coalitions are restricted to $k$ or fewer agents. We present two different algorithms. Each guarantees the generated solution to be within a constant factor, for fixed $k$, of the optimal in terms of utility. Our algorithms complement Shehory and Kraus' algorithm (Artif Intell 101(1–2):165–200, 1998), which provides guarantee's on solution cost, as ours provides guarantees on utility. Our algorithm for general multi-agent domains is a modification of and has the same running time as Shehory and Kraus' algorithm, while our approach for multi-robot domains runs in time $O(n^{\frac{3}{2}}m)$, much faster than Vig and Adams (J Intell Robot Syst 50(1):85–118, 2007) modifications to Shehory and Kraus' algorithm for multi-robot domains, which ran in time $O(n^k m)$, for $n$ agents and $m$ tasks.

**Keywords** Coalition formation · Task allocation · Multi-robot

T. C. Service (✉)
Vanderbilt University, 357 Jacobs Hall, 2201 West End Nashville, Nashville, TN 37240, USA
e-mail: tservice@acm.org

J. A. Adams
Vanderbilt University, 359 Jacobs Hall, 2201 West End Nashville, Nashville, TN 37235-1824, USA
e-mail: julie.a.adams@vanderbilt.edu

## 1 Introduction

Due to the inherent difficulty and complexity of real world tasks, cooperation amongst autonomous agents is essential for successful task completion. Of interest is how to best utilize a set of heterogenous agents to complete a set of tasks. One aspect of the task allocation problem is what is known as the coalition formation problem: How do we form teams of agents, with each team assigned to a particular task, in order to best complete the set of tasks at hand?

While the coalition formation problem in multi-agent systems has received much attention [4,12,14,16], only a small fraction of that work has focused on multi-robot domains [2,5,15,17,20]. The multi-agent algorithms and techniques are not directly transferable to multi-robot domains, as multi-robot domains present challenges and constraints not encountered with software agents [19]. Further, much work in multi-agent coalition formation has made use of characteristic function games [14], which assume each coalition is assigned a value independent of the task they are assigned. For domains where we are interested in forming coalitions for task allocation, the value of a particular coalition, in general, depends on the task it is assigned to complete.

The focus of this paper is on coalition formation for task allocation in both general multi-agent domains as well as multi-robot domains. In order to accommodate both domains we consider two different formalizations of the coalition formation problem for task allocation, one originally designed for multi-agent domains where agent resources are transferable and one designed for multi-robot domains. While it has been shown that it is NP-hard to approximate the optimal coalition structure in characteristic function games to within a factor of $O(c^{1-\epsilon})$ for any $\epsilon > 0$, where $c$ is the number of non-zero valued coalitions [14], such results do not immediately apply to task allocation. This paper demonstrates a similar inapproximability result in terms of the number of tasks for formalizations of the coalition formation problem for both the multi-agent and multi-robot domains.

In light of these inapproximability results we turn our attention to two natural restrictions of the coalition formation problem. We first consider the situation where each agent in the system is one of a fixed number of $j$ different types. Agents of each type are indistinguishable from one another (e.g., in a multi-robot domain we may have a set of identical unmanned arial vehicles (UAVs) as well as a set of identical unmanned ground vehicles (UGVs) with which to complete our tasks). For this situation, we present a dynamic programming based approach, which, for a fixed $j$ finds the optimal coalition structure in polynomial time and is applicable in both multi-agent and multi-robot domains.

The second situation we consider is where the size of the allowed coalitions are bounded above by a fixed constant $k$. This case was previously studied by Shehory and Kraus [16] who developed a greedy iterative algorithm based on the set covering and set partitioning problem, which is guaranteed to find a coalition structure of total cost[1] within a logarithmic factor of the optimal [16]. However, their algorithm does not apply directly to multi-robot domains [19]. Previous attempts to use Shehory and Kraus' algorithm in multi-robot domains have added the overhead of a constraint satisfaction problem in order to enforce the physical constraints on physical robot resources [19].

We provide a discussion on the differences between minimizing overall coalition cost versus maximizing overall utility. We show that depending on how cost is defined in terms of utility, the relationship between coalition structure cost and coalition structure utility can be complex and that the lowest cost coalition structure is not necessarily the highest utility

[1] The cost of a coalition structure is defined as the sum of the reciprocals of the utilities of the satisfied tasks.

structure. We also demonstrate a reduction from the $k$-set packing problem to the coalition formation problem showing that no algorithm can in polynomial time, unless $P = NP$, be guaranteed to find a coalition structure that is within a factor of $\Omega(k/\log k)$ of the optimal. Thus, any algorithm to generate approximate solutions in polynomial time must have a worst case ratio bound between the utility of the generated solutions and the utility of the highest valued solution that grows almost linearly with the size of the allowed coalitions. This result demonstrates that, while Shehory and Kraus' algorithm is guaranteed to produce a solution with total cost within a logarithmic factor of the lowest cost solution it cannot, in general, produce a solution with utility within a logarithmic factor of the highest utility coalition structure. We then prove a stronger result showing that no approximation algorithm, whose performance ratio is a function of only the size of the problem instance (i.e., number of agents and tasks) can simultaneously guarantee any finite bound on both the utility and the cost of the solution that it finds.[2]

This paper presents two algorithms for the situation where coalitions are restricted to $k$ or fewer agents. The first is a modification of Shehory and Kraus's algorithm, which is guaranteed to find a coalition structure with utility within a factor of $k + 1$ of the optimal, for a fixed $k$ and has the same computational complexity, $O(n^k m))$, for $n$ agents and $m$ tasks. Like Shehory and Kraus's algorithms, our algorithm is an anytime algorithm that improves the quality of its solution monotonically as time progresses. The second algorithm is based on a service oriented task allocation model used previously by Vig and Adams [20] for multi-robot domains. Our algorithm for the multi-robot domain guarantees the same bound as our algorithm for the general multi-agent domain, but reduces the computational complexity from $O(n^k m))$ to $O(n^{\frac{3}{2}} m)$. Previous attempts to employ a greedy iterative technique, like that of Shehory and Kraus' algorithm in multi-robot domains have run in $O(n^k m))$ time and incurred the additional overhead of a constraint satisfaction problem [19]. Our algorithm for the multi-robot domain, thus, scales to larger allowed coalition sizes far better than the previous attempts to employ Shehory and Kraus's algorithm in multi-robot domains. The reduction from $k$-set packing to coalition formation for task allocation shows that any polynomial time approximation algorithm for coalition formation must have an approximation ratio that grows almost linearly with the size of the allowed coalitions. Our approximation ratio is, thus, almost asymptotically tight as it grows linearly with the size of the allowed coalitions.

The remainder of this paper is structured as follows. Section 2 provides the necessary background and surveys the related work on coalition formation for task allocation. Section 3 presents the formal problem definitions considered in this paper. Section 4 presents complexity theoretic results demonstrating some differences between the two formalizations of the coalition formation problem and proves that not only is the coalition formation problem NP-hard to solve exactly, it is also NP-hard to approximate within a factor of $O(m^{1-\epsilon})$ for all $\epsilon > 0$, where $m$ is the number of tasks. Algorithms and complexity results for two natural restrictions of the coalition formation problem are presented in Sect. 5. An empirical validation of our algorithms is presented in Sect. 6, which is followed by concluding remarks in Sect. 7.

[2] This result uses the explicit definition of the cost of a coalition as the reciprocal of its utility, as used by Shehory and Kraus [16].

## 2 Related work

Much work has gone into task allocation and coalition formation in multi-agent systems [1,11,14,16,18–20]. Both theoretical results [1,14] and practical algorithms [16,20] have been presented.

Many researchers have pointed out the computational difficulty of a variety of formalizations of the coalition formation problem [1,14,16]. Sandholm et al. [14] take a game theoretic approach, viewing the coalition formation problem as a characteristic function game. Each possible coalition $S$ (subset of agents) is assigned a value $v_S$. A coalition structure $CS$ is then simply a partitioning of the agents. The goal is to find the coalition structure $CS^*$ that maximizes the sum of the coalition values:

$$CS^* = \arg\max_{CS} \sum_{S \in CS} v_S.$$

Sandholm et. al. [14] show that finding the optimal coalition structure is an $NP$-hard problem. They go on to show that in order to obtain any bound on the solution quality, in characteristic function games a search algorithm is required to visit $O(2^{n-1})$ coalition structures, where $n$ is the number of agents. They provide an algorithm that examines those coalition structures that partition the agents into one or two coalitions, which is guaranteed to return a solution within a factor of $n$ from the optimal. The requirement of $O(2^{n-1})$ examined coalition structures results from the fact that coalitions can be given arbitrary values. In order to be within a guaranteed bound of the optimal solution, every coalition must be present in at least one of the examined coalition structures, as an unexamined coalition can potentially have an arbitrarily high value.

An issue with using characteristic function games when modeling coalition formation for task allocation is that characteristic function games do not account for the tasks to which each coalition is assigned. That is, the value of a coalition $S$ is independent of everything other then its members. This is unrealistic for task allocation settings. A particular coalition $S$ may be very well suited for, and thus perform well on, a particular task $t$, but may be very poorly suited for another task $t'$. The value of $S$; therefore, depends directly on the task that $S$ is to perform.

Both Shehory and Kraus [16] and Abdallah and Lesser [1] consider coalition formation for the task allocation problem and demonstrate that finding the optimal coalition structure in this formalization is $NP$-hard. Due to the computational intractability of the coalition formation problem, both Shehory and Kraus [16] and Abdallah and Lesser [1] present heuristic approaches to coalition formation. Shehory and Kraus' algorithm is particularly relevant for our work and is discussed separately later.

In light of $NP$-hardness results for coalition formation, many of the algorithms developed for coalition formation are anytime algorithms [12,14,16], which permit the search for an optimal coalition structure to be terminated at any point while still providing a solution. Anytime algorithms permit the agents to dynamically decide whether using additional search time is worth the possibility of an increase in coalition structure utility. Another advantage of anytime algorithms is that under time constraints anytime algorithms can at least produce a solution, whereas non-anytime algorithms may not have sufficient time to generate a solution.

Many of the theoretical results concerning the hardness of the coalition formation problem point out relationships between the coalition formation problem and many classical problems in complexity theory. For example, Shehory and Kraus [16] discuss the relationship between both the set covering problem and set partitioning problem and the coalition formation

problems. Abdallah and Lesser [1] demonstrate that the multi-dimensional knapsack problem is reducible to the coalition formation problem.

While significant research exists for multi-agent coalition formation [4,12,14,16], only a small fraction of research has focused on multi-robot domains [2,5,15,17,20]. Vig and Adams [19,20] demonstrate some of the differences between general multi-agent and multi-robot domains and highlight the fact that the general multi-agent algorithms and techniques are not directly transferable to multi-robot domains, as multi-robot domains present challenges and constraints not encountered with software agents. Vig and Adams [20] introduce the service oriented model employed in this paper and present a market based approach, RACHNA that attempts to address some of the issues with coalition formation in multi-robot domains. The RACHNA system leverages redundancies in agents. Each agent is modeled as being able to perform a set of services and each task requires a certain amount of each type of service. Each task is assigned a fixed amount of utility and the tasks bid on the services they require in a multi-unit combinatorial auction.

While RACHNA was shown to be capable of producing high quality coalition structures, Sandholm [13] show, via a reduction from the maximum weighted independent set problem, that, unless $P = NP$,[3] it is impossible to approximate the winner of a combinatorial auction in polynomial time to within a bound of $O(b^{1-\epsilon})$ for any $\epsilon > 0$, where $b$ is the number of bids. This result; however, does not imply that approximating the optimal coalition structure is difficult. It simply suggests that techniques employing general combinatorial auctions to solve the coalition formation problem cannot achieve such bounds in polynomial time. However, in Sect. 4 we provide a proof that the general coalition formation problem cannot be approximated within a factor of $O(m^{1-\epsilon})$ for any $\epsilon > 0$, where $m$ is the number of tasks, unless $P = NP$.

Campbell et al. [4] consider a different aspect of task allocation in multi-agent systems. They consider the situation where the group of agents is presented with a sequence of tasks. It is up to each individual agent to decide whether or not to join the coalition to solve each task. They present a dynamic approach, which continually updates an agents "tolerance" for tasks (i.e., the difficulty of a task compared to the ability of the agent). Compared to both a greedy and a fixed tolerance based approach they find that, under certain conditions on the distribution of task difficulties and lengths, their dynamic approach minimizes the amount of time required to complete the tasks. An additional benefit of the dynamic tolerance approach they present is that, unlike the greedy method, it requires no explicit communication between agents, permitting near perfect scaling, and is robust to changes in the agents.

A classification of coalition formation problems is given by Lau and Zhang [11]. They consider five distinct classes of coalition formation problems based on three axes: job characteristics, resource constraints and the objective function. For each case they present algorithms and complexity results. The situation they present considered overlapping coalitions. However, in multi-robot situations tasks may potentially be geographically distant, or distant enough to not permit robots to work on two tasks simultaneously, and as such coalitions may need to be pairwise disjoint.

Two previously suggested heuristic based coalition formation algorithms are highly relevant to this paper. Both Shehory and Kraus' algorithm [16] for coalition formation in general multi-agent domains and Vig and Adams' extension to Shehory and Kraus' algorithm [19,20] for multi-robot domains are discussed separately.

[3] The theorem given by Sandholm [13] states the inapproximability result for $NP \neq ZPP$; however, this can be strengthened to $P \neq NP$ using the results of Zuckerman [21].

### 2.1 Shehory and Kraus's algorithm

Shehory and Kraus [16] modeled coalition formation for the task allocation problem in terms of the resources required by tasks and provided by agents. They presented a heuristic based coalition formation algorithm where a limit was imposed on the maximum size of a coalition, $k$. They presented an iterative greedy algorithm that runs in $O(n^k m))$ time, where $n$ is the number of agents and $m$ is the number of tasks. They also prove a logarithmic performance bound of the cost of the returned coalition structure from the optimal, where the cost of a coalition is defined by the reciprocal of its value. However, their result does not imply a logarithmic performance bound on the total utility of the coalition structure. The results presented in this paper show that a logarithmic performance bound on the utility of the generated structure cannot be achieved, in general, in polynomial time.

Shehory and Kraus' original greedy algorithm is based upon approximation algorithms for the set covering and set partitioning problem and proceeds in two stages:

1. Distributive calculation of coalition values.
2. A distributed greedy algorithm iteratively selects the coalition to include in the solution.

Algorithm 2.1 shows the preliminary stage of Shehory and Kraus' algorithm. The agents distributively agree on which coalitional values each agent will calculate. The variable $P_i$ is the set of all potential coalitions that contain $a_i$ as a member, $S_{ij}$ is the set of coalitions that contain both agent $a_i$ and agent $a_j$ and $L_i$ is a list of potential coalitions for which agent $a_i$ has agreed to repeatedly calculate the values.

**Algorithm 2.1** Preliminary Stage

1: Form a list $P_i$ of all coalitions of up to $k$ agents which include agent $a_i$
2: **while** $P_i \neq \emptyset$ **do**
3: Contact an agent $a_j$ who is a member of some coalition in $P_i$.
4: Agree to the calculation of a subset $S_{ij}$ of $P_i$ of coalitions of which both $a_i$ and $a_j$ are a member.
5: Add $S_{ij}$ to $L_i$, the list of coalitions $a_i$ has agreed to repeatedly compute values for.
6: Remove $S_{ij}$ from $P_i$.
7: Remove $S_{ki}$ from $P_i$ for each agent $a_k$ that contacts $a_i$.
8: Compute values for the coalitions in $S_{ij}$. The value of a coalition is the maximum utility it can obtain by completing any task.
9: **end while**

Define the weight, $w_i$, of a coalition $C_i$, as: $w_i = \frac{c_i}{|C_i|}$, where $c_i$ is the cost of coalition $C_i$. Once the initial coalition values have been calculated, the second stage of the algorithm iteratively constructs a coalition structure. Algorithm 2.2 provides psuedo-code for the second stage of Shehory and Kraus' algorithm.

**Algorithm 2.2** Shehory and Kraus's Algorithm

1: **while** $\mathcal{T} \neq \emptyset$ and $L_i \neq \emptyset$ **do**
2: Find the coalition $C_j \in L_i$ with the smallest weight.
3: Inform all agents of the coalition weight found.
4: Select the coalition and task pair, $C_{low}$ and $t_{low}$, of the lowest broadcasted weight.
5: $\mathcal{T} \leftarrow \mathcal{T} - \{t_{low}\}$
6: $L_i \leftarrow L_i - C_{low}$
7: Recalculate the coalition values of coalitions in $L_i$ that require recalculation.
8: **end while**

We present a modification to Shehory and Kraus' algorithm that guarantees a solution with total utility within a factor of $k + 1$ from the optimal. Our algorithm, thus, provides a guaranteed bound on the total solution utility rather than the solution cost. We show in Theorem 5.1 that while Shehory and Kraus [16] show a logarithmically growing approximation ratio, in terms of the size of the allowed coalitions, between the cost of the solution generated by their algorithm and that of the minimum cost coalition structure, this bound cannot apply in general for the ratio between the utility generated by their algorithm and the maximum utility coalition structure, unless $P = NP$.

### 2.2 Vig and Adams' algorithm for multi-robot domains

Vig and Adams [19,20] consider the coalition formation problem in multi-robot domains. In such situations the individual coalition member's resources are not necessarily transferable. Vig and Adams present a service oriented model, where each agent or robot in the coalition performs a specific service in order to complete the task at hand [20]. They empirically validate an extended version of Shehory and Kraus's algorithm in multi-robot domains that includes an additional constraint satisfaction problem in order to restrict the distribution of resources amongst agents (e.g., performing a box pushing service requires both a camera and pusher on a single robot).

Vig and Adams [20] highlight the point that software agent and robot domains differ in several key ways:

1. Software agent resources correspond to fragments of code or data; while, robot resources correspond to physical sensors and actuators located on the robot.
2. Software agent resources are typically transferable; while, robot resources (e.g., sensors and actuators) are not.
3. Software agents cooperate through the exchange of information; while, robots cooperate through both information exchange and their effects on the physical world.
4. Software agents operate in environments that are typically free of real-world constraints; physical robots do not.

An important consequence of the differences between software agents and physical robots is that for a coalition of robots to be capable of completing a particular task, certain constraints on resource locations must be met. A coalition of robots simply possessing the necessary resources to complete a task does not necessarily imply that the coalition can complete the task, since in multi-robot domains resources usually are not transferable between coalition members. Coalition formation in multi-robot domains is further complicated by the inherent uncertainty of real-world domains. Due to these differences (and others [20]) between software agent and robot domains, many of the coalition formation algorithms developed for general multi-agent domains are not directly applicable to multi-robot domains [20]. The resource model employed in much of the previous coalition formation work [1,16,18] is, thus, not appropriate for multi-robot domains where resources are not transferable.

Vig and Adams model the physical resource constraints inherent in multi-robot domains as a constraint satisfaction problem and employ the arc consistency algorithm to verify that a given coalition can feasibly complete a given task. While Vig and Adams demonstrated that this heuristic approach works in multi-robot domains, their approach incurs the additional overhead of solving a constraint satisfaction problem (although it did not affect the computational complexity of their algorithm). Also, the logarithmic performance ratio on the total solution cost guaranteed by Shehory and Kraus's algorithm does not necessarily hold under

the extensions of Vig and Adams, as Vig and Adams' extensions prevent certain coalition task pairs from being allocated due to additional constraint violations.

After extending Shehory and Kraus' algorithm for the multi-robot domains, Vig and Adams [20] presented a new formalization of the coalition formation problem for multi-robot domains where agents are assigned to perform a specific service for the task to which they are assigned. Vig and Adams' service model abstracts away the individual resources each agent possesses to the set of services the agents can perform and removes the need for verifying resource constraints to ensure coalition task pair feasibility. We improve on Vig and Adams' work by presenting a new algorithm that directly incorporates the notion of services and guarantees a solution within a factor of $k+1$ from the optimal in $O(n^{\frac{3}{2}}m)$, for any $k$. This is much faster than the Vig and Adams' algorithm, which ran in time $O(n^k m)$). Further, our algorithm scales much better than Vig and Adams', as the complexity of their approach increases exponentially with $k$, while the asymptotic complexity of our algorithm is independent of $k$.

## 3 Problem description

Two different formulations of the coalition formation problem are considered in this paper, one previously used in general multi-agent domains and one designed for multi-robot domains. For both definitions, let $\mathcal{A}$ be a set of $n$ agents and $\mathcal{T}$ be a set of $m$ tasks. In both formalizations, it is assumed that the agents are group rational and attempt to maximize the total utility gained by the group. We consider only non-overlapping coalitions.

The first formalization presented in Definition 1 has been used previously in multi-agent domains [1,16,18].

**Definition 1** (Resource Model) Each task $t \in \mathcal{T}$ has an associated utility $u_i$ representing the value that completing the task is worth and a vector $R_t = (r_1, r_2, \ldots, r_p)$ indicating the quantity of each type of resource it requires (i.e., task $t$ requires $r_i$ units of resource $i$). Associated with each agent $a$ is a vector of resources $R_a = (r_1, r_2, \ldots, r_p)$ indicating the quantity of each resource that agent $a$ can provide (i.e., agent $a$ can provide $r_i$ units of resource $i$). Given an assignment of agents to tasks, with each agent assigned to at most one task, a task $t$ is satisfied if and only if the sum of each of the individual resources provided by agents assigned to $t$ is at least the amount required by $t$.

**Definition 2** (Service Model) Each task $t \in \mathcal{T}$ has an associated utility $u_i$ representing the value that completing the task is worth and a vector $S_t = (s_1, s_2, \ldots, s_p)$ indicating the quantity of each type of service it requires (i.e., task $t$ requires $s_i$ agents to perform service $i$). Associated with each agent $a$ is a vector of services $S_a = (s_1, s_2, \ldots, s_p)$ indicating the possible services that agent $a$ can provide (i.e., agent $a$ can perform service $i$ if and only if $s_i$ is 1). An assignment of agents to tasks must specify the service that each agent will provide. Each agent can be assigned to perform only a single service. Given an assignment of agents to tasks and corresponding services, with each agent assigned to at most one task, a task $t$ is satisfied if and only if every service required by that task is performed by some agent.

The service model was introduced by Vig and Adams [20] in order to enforce resource constraints within tasks. Resources in multi-robot domains often correspond to robot actuators and sensors and as such are often not autonomously transferable between robots. One can view the service model as an instance of the resource model along with resource constraints. For example, an agent may be capable of performing the box pushing service if and only

if it has both a laser range finder resource (used to find and align itself with the box) and a manipulator resource (used to hold on to the box). As done by Vig and Adams [20], one can model such multi-robot domains with the resource model along with a set of resource constraints as a constraint satisfaction problem for each task; however, the service model, as introduced by Vig and Adams [20] abstracts such details away.

While the theoretical results and algorithms presented in this paper apply equally well to both problem definitions, we will employ the terminology of the service model (Definition 2) in the theorem proofs and algorithm descriptions. When necessary, we will describe the changes to show how the results and algorithms apply to the the resource model definition.

It should be noted that Shehory and Kraus [16] define the utility gained by completing a task as a function of the resources that task required. However, this definition is restrictive as one can imagine high valued tasks that require few resources (e.g., moving an injured victim to safety in a first response situation is an important task that does not necessarily require many resources). We instead consider the more general situation where tasks may be a priori assigned arbitrary utilities.

## 4 Complexity results

While the resource and service models are very similar, the complexities of answering even simple questions varies greatly. For example, a natural question one might ask is given a single task, what is the smallest coalition of agents that can complete the given task? This question is NP-hard to solve in general under the resource model, but can be solved easily in $O(n^{\frac{5}{2}})$ time under the service model.

**Theorem 4.1** *Determining the smallest coalition of agents capable of completing a single task is NP-hard under the resource model (Definition 1), but is solvable in $O(n^{\frac{5}{2}})$ time under the service model (Definition 2).*

*Proof* We reduce the set covering problem to an instance of the coalition formation problem under the resource model. Given a collection $C$ of $l$ finite sets $S_1, \ldots, S_l$ define the set of all possible resources to be $R = \cup_{i<l} S_i$ and define a task $t$ that requires a single unit of each resource. For each set $S_i$, define an agent $a_i$ that has a single unit of each resource in $S_i$. A coalition of agents capable of completing task $t$ is then a set cover of $C$ and a minimum size coalition capable of completing task $t$ is then a minimum set cover of $C$.

For the service model, we reduce the problem of finding a smallest coalition of agents capable of completing a single task $t$ under the service model to an instance of the bipartite matching problem. Let $k$ be the number of services required by task $t$ (including multiplicities, e.g., $t$ requires $x$ agents performing service $i$). If $t$ requires more services then there are agents, then no coalition can satisfy $t$. Otherwise, for each agent $a_i$ we define a vertex $v_i^a$ and for each service $s_j$ required by the task $t$ (including their multiplicities) we define a vertex $v_j^s$. An edge is added between vertex $v_i^a$ and vertex $v_j^s$ if and only if $a_i$ is capable of performing service $s_j$.

A matching in this graph is a set of pairwise disjoint edges (i.e., no two edges share a common vertex as an end point). If there is a coalition of agents capable of performing the task $t$, then there is a maximum matching of size $k$ in the constructed graph (since every service must be matched with an agent capable of performing that service). A coalition that satisfies the task $t$ can be formed from such a matching by assigning agent $a_i$ to perform service $s_j$ for task $t$ whenever the edge $(a_i, s_j)$ is in the matching. Since $k \leq n$, there are $n + k = O(n)$ ver-

tices and at most $nk = O(n^2)$ edges. Using the Hopcroft and Karp algorithm [9], a maximum matching can be found in $O(n^{\frac{5}{2}})$ time.

We can also consider weighted variants of this question by assigning a weight to each agent and attempting to find a minimum weight coalition that can complete a given task $t$. In this situation, the weights can, for example, be a function of the capabilities of the agents (e.g., the sum of the resources that agent has or the number of services it can provide) and the minimal weight coalitions are those with the minimal amount of total capabilities that can complete the task $t$, thereby maximizing the total capabilities of the remaining agents. A similar result holds for the weighted situation.

**Theorem 4.2** *Determining the smallest weight coalition of agents capable of completing a single task is NP-hard under the resource model (Definition 1), but is solvable in $O(n^4)$ time under the service model (Definition 2).*

*Proof* Clearly this problem is NP-hard for the resource model, as it is a generalization of the unweighted case.

The same reduction as was used in the proof for the service model case in Theorem 4.1 can reduce this problem to an instance of the weighted bipartite matching problem. An edge $(v_i^a, v_j^s)$ is defined with a weight equal to the negation of the weight of $a_i$ plus a constant so that all edge weights are positive, if $a_i$ can perform service $s_j$. Since the maximum weight bipartite matching problem can be solved in $O(|V|^2|E|)$ time [10], we can determine the minimum weight coalition capable of solving the task $t$ in $O(n^4)$ time.

In this setting and throughout the remainder of this paper we will consider a reasonable approximation factor to be one that grows significantly less than linear in number of tasks. Formally, we define a reasonable factor to be one that is $O(m^{1-\epsilon})$ some $\epsilon > 0$, where $m$ is the number of tasks. A natural question to consider when attempting to solve NP-hard problems is whether or not they can be approximated to within a reasonable factor. We provide a negative answer to this question for the coalition formation problem. While previous work has shown similar results for coalition formation in characteristic function games, this is a new result for the resource and service models (Definitions 1 and 2).

**Theorem 4.3** *Both the resource and service models of coalition formation (Definitions 1 and 2) cannot be approximated within a factor of $O(m^{1-\epsilon})$, where $m$ is the number of tasks, for any $\epsilon > 0$, unless $P = NP$.*

We will reduce the maximum independent set problem to the coalition formation problem from which the theorem will follow immediately. The reduction presented reduces the maximum independent set problem to the service model. We will only briefly describe the reduction to the resource model; however, the reduction is identical to the reduction under the service model when the services are replaced by resources.

*Proof* Let $G = (V, E)$ be an arbitrary graph and let $m$ and $n$ denote the number of vertices and edges, respectively.[4] For each vertex $v_i$ define a task $t_i$ and for each edge $e_i$ define both an agent $a_i$ and a service $s_i$. For each agent $a_i$ corresponding to an edge $e_i$ we define $a_i$'s service vector to consist of all 0's except for the $i$-th service (i.e., each agent $a_i$ can perform only

[4] Traditionally, the number of vertices has been denoted by $n$ and the number of edges denoted by $m$; however, to remain consistent with the coalition formation literature, and since we are mapping vertices to tasks, we have opted to denote the number of vertices by $m$ and the number of edges by $n$.

service $s_i$). For each edge incident to a node $v_i$ add that service to the requirements vector of task $t_i$. Each task $t_i$ is then satisfied only when the agents corresponding to the edges incident to $v_i$ are assigned to $t_i$. Thus, two tasks $t_i$ and $t_j$ can be simultaneously satisfied if and only if the corresponding vertices in $G$, $v_i$ and $v_j$, are not connected by an edge. Define the utility of each task to be 1.

Now consider any independent set $I$ in $G$, $I$ corresponds naturally to a coalition structure. For each edge $e_k$ incident to a node $v_i \in I$, assign agent $a_k$ to task $t_i$. The utility of such an assignment is simply the number of satisfied tasks, which is $|I|$. Similarly, each assignment of agents to tasks corresponds to an independent set in $G$. For each satisfied task $t_i$ in the coalition structure, add the vertex $v_i$ to the set $I$.$I$ must form an independent set, as for a task $t_i$ to be satisfied it must be allotted each agent corresponding to an edge incident to $v_i$ in $G$ and, thus, none of $v_i$'s neighbors in $G$ can correspond to satisfied tasks in the coalition structure. This reduction can clearly be done in polynomial time.

Thus, there is a one-to-one mapping between the independent sets of $G$ and coalition structures, such that if an independent set in $G$ has $l$ vertices, the utility of the corresponding coalition structure is $l$. An approximation algorithm to the coalition formation problem then provides an approximation algorithm for the maximal independent set problem. Based on Zuckerman's results [21] it can be concluded that the coalition formation problem is NP-hard to approximate to within a factor of $O(m^{1-\epsilon})$ for all $\epsilon > 0$.

The reduction used in the proof of Theorem 4.3 also applies to the resource model (Definition 1). Instead of each edge corresponding to an agent and a service, each edge corresponds to both an agent and a resource. Agent $a_i$ corresponding to edge $e_i$ has only a single unit of resource $r_i$. Task $t_k$ requires a unit amount of resource $r_i$ if and only if edge $e_i$ is incident to vertex $v_k$ in $G$. Thus, any two tasks $t_i$ and $t_j$ are simultaneously satisfiable if and only if $v_i$ and $v_j$ do not share an edge in $G$. The rest of the proof follows naturally.

Note that we can trivially find an approximate solution to the coalition formation problem within a factor of $O(m)$ in polynomial time by simply assigning all agents to the highest utility task. Further, it has been conjectured that the best performance ratio achievable for the independent set problem is $O(m/polylog(m))$ (where again $m$ is the number of vertices) [6]. If this conjecture is true then, this lower bound on the possible performance ratio carries over directly to the coalition formation problem.

While the general problem of coalition formation is NP-hard even to approximate within a reasonable factor, natural restrictions to the problem can be solved exactly in polynomial time.

## 5 Restricted classes of coalition formation problems

While we cannot hope to solve or approximate, within a reasonable factor, the coalition formation problem in polynomial time (unless of course $P = NP$), we consider natural restrictions of the coalition formation problem of practical interest that can be either solved or approximated efficiently.

We begin by considering the situation in which there are a fixed $j$ types of agents and each agent of a particular type provides the same resources or can perform the same services as all agents of that type. We provide a dynamic programming based algorithm that solves the coalition formation problem for this situation in polynomial time for each fixed $j$.

We also consider the same situation as considered by Shehory and Kraus [16], where the maximum size of any allowed coalition is bounded above by a constant $k$. Shehory and

Kraus' algorithm [16] is based upon approximation algorithms for the set covering problem and attempt to minimize the total cost of the coalition structure, defined to be the sum of the reciprocals of the utilities of the satisfied tasks. We present algorithms that complement Shehory and Kraus' algorithm for general multi-agent domains and improve upon Vig and Adams' algorithm [19] in the following ways:

1. our algorithms are guaranteed to produce a solution within a constant bound of the optimal, for a fixed $k$, in terms of total utility rather than cost, and
2. for the service model we substantially improve the running time, from the $O(n^k m)$ runtime of Vig and Adams' algorithm [19] to $O(n^{\frac{3}{2}} m)$, thus improving the scalability to larger allowed coalition sizes.

The first contribution is important, since minimizing the total coalition structure cost is not the same as maximizing the total utility, as we will demonstrate. Previous attempts to use Shehory and Kraus' algorithm in multi-robot domains required the addition of a constraint satisfaction problem in order to ensure that particular sets of resources reside on particular robots [19]. The second contribution is also important, because it eliminates this need and results in an algorithm that has a significantly reduced computational complexity. The algorithm we present for the service model also scales exceptionally well, since the computational complexity is independent of $k$; whereas, Vig and Adams' extension of Shehory and Kraus' algorithm scales exponentially with $k$ [16,19]. Further, it is unknown whether the original bound on the coalition structure cost guaranteed by Shehory and Kraus' original algorithm applies with the CSP modifications by Vig and Adams. Our presented algorithm provides a guarantee on the total utility for both the multi-agent and multi-robot domains.

## 5.1 Bounded number of agent types

We first consider the situation in which each agent is drawn from one of a fixed set of $j$ types, where each agent of a given type has identical capabilities (e.g., can perform the same services or has the same resources). We present a dynamic programming based algorithm that finds an optimal coalition structure in polynomial time for each fixed $j$. These algorithms are very appropriate for multi-robot domains where each robot can perform some subset of a fixed set of $s$ services. In this situation, the number of possible combinations of services an agent can perform is $2^s - 1$, a constant for fixed $s$, resulting in a constant number of agent types.

### *5.1.1 Homogeneous agents*

Arguably the most natural starting point for considering restricted instances is the case in which all of the agents are identical (i.e., $S_i = S_j$, $R_i = R_j$, for all pairs of agents $a_i, a_j$). In this situation, the agents can be viewed as a discrete finite resource that must be divided among the tasks. For each task, $t_i$ there is a minimum number of agents, $r_i$, required to complete the task. Algorithm 5.1 provides pseudo-code for the $O(nm)$ dynamic programming based approach to coalition formation for homogeneous agents.

Algorithm 5.1 constructs an $n \times m$ table, where the $(i, a)-$th entry contains the maximum possible utility of the subproblem consisting of only $a$ of the homogeneous agents and only tasks $i, \ldots, m$. It is easy to see that Algorithm 5.1 generates the optimal coalition structure in $O(nm)$ time, assuming the number of resources or services is $O(n)$.

**Algorithm 5.1** Coalition Formation For Homogeneous Agents

```
for a ∈ {1, ..., n} do
  if a ≥ r_m then
    Table_{m,a} = u_m
  else
    Table_{m,a} = 0
  end if
end for
for i = m − 1 to 1 do
  for a ∈ {1, ..., n} do
    if a ≥ r_i then
      Table_{i,a} = MAX(Table_{i+1,a−r_i} + u_i, Table_{i+1,a})
    else
      Table_{i,a} = Table_{i+1,a}
    end if
  end for
end for
```

### *5.1.2 Heterogeneous agents*

In the situation where each agent is drawn from one of $j$ fixed types, each agent type can be viewed as a different resource. Since there are $n$ agents, the number of agents of each type is $O(n)$. There are only $O(n^j)$ unique coalitions (i.e., specifying $0 \leq n_i \leq n$ agents of type $i$ for each type). We extend Algorithm 5.1 for use in situations with heterogeneous agents by extending the table to include a dimension for each agent type. The table is, thus, of size $O(n^j m)$ and determining the value of each entry in the table requires $O(n^j)$ time since there are $O(n^j)$ unique coalitions. The total running time is $O(n^{2j} m)$.

## 5.2 Bounded coalition size

We now consider the same situation that Shehory and Kraus [16] considered where the size of the allowed coalitions are bounded above by a constant $k$. This is a particularly natural restriction in the multi-robot domain as often tasks require less than a fixed number of robots [20]. Specifically, for the service model, this restriction fits naturally when all tasks require less than $k$ services (including their multiplicities). For this situation, we present algorithms with nearly asymptotically optimal performance ratios. The theoretical results and performance bounds presented in this sect. are with respect to the greatest utility coalition structure consisting of coalitions of $k$ or fewer agents.

Our algorithms differ from Shehory and Kraus' algorithm [16] in that they provide guarantees on the total utility of the solution they generate instead of providing guarantees on the total cost. We show that, in general, for certain definitions of coalition cost, that maximizing utility and minimizing cost are not the same. As such, our algorithms can be viewed as complementary to Shehory and Kraus' algorithm, since our algorithms are appropriate for domains in which maximum utility is desired; whereas, Shehory and Kraus' algorithm is appropriate for domains in which minimum cost is desired.

### *5.2.1 Inapproximability results*

We first provide a bound on the approximation ratio an algorithm can obtain in polynomial time for finding coalition structures of size at most $k$. We will use this bound to show that the guarantee our algorithm provides is almost asymptotically tight.

**Theorem 5.1** *Both the service and resource models, when coalition size is restricted to k or fewer agents, are NP-hard to approximate within a factor of* $\Omega(k/\log(k))$.

*Proof* Reduce the $k$-set packing problem to this restricted version of the coalition formation problem. Given a collection, $C$, of sets $S_1, \ldots S_l$, the set packing problem is to find a maximum cardinality subset $C'$ of $C$ whose elements are pairwise disjoint. It has previously been shown that the $k$-set packing problem, where each set $S_i$ has no more than $k$ elements, cannot be approximated within a factor of $\Omega(k/\log(k))$, unless $P = NP$ [8].

Given a collection $C$ of sets $S_1, \ldots, S_l$, define an instance of the coalition formation problem as follows. Define the set of agents $A$ to be the union of all the sets $S_1, \ldots, S_l$. For each set $S_i$ define a task $t_i$, such that task $t_i$ has utility 1 and task $t_i$ requires $|S_i|$ agents to perform service $s_i$ (alternatively $t_i$ requires $|S_i|$ units of resource $r_i$). Each agent in the set $S_i$ is capable of performing service $s_i$ (provides 1 unit of resource $r_i$) and every agent not in $S_i$ is incapable of performing service $s_i$ (provides 0 units of resource $r_i$). Thus, the only way for a task $t_i$ to be satisfied is for all agents in $S_i$ to be assigned to it.

A set packing consisting of $j$ disjoint sets then corresponds to a coalition structure of total utility $j$, as each agent in a set $S_i$ in the packing can simply be assigned to task $t_i$. Likewise, a coalition structure of total utility $j$ corresponds to a set packing consisting of $j$ disjoint sets, since each set $S_i$ can be added to the packing if and only if task $t_i$ is satisfied. Approximating the coalition formation problem to within any factor thus corresponds to approximating the original set packing problem to within the same factor. Since the $k$-set packing problem is $NP$-hard to approximate within a factor of $\Omega(k/\log(k))$, so are both the service and resource models of the coalition formation problem, when coalitions are restricted to $k$ or fewer agents.

Thus, in the best case, the approximation ratio obtained by a polynomial time approximation algorithm for the coalition formation problem, when restricting the size of the coalitions to at most $k$ agents, must grow almost linearly with $k$. This implies that the low logarithmic ratio bound, in terms of cost, provided by Shehory and Kraus' algorithm does not apply to the utility of the solutions generated by their algorithm, unless of course $P = NP$, as a bound of $\log k$ is asymptotically bounded above by $k/\log k$.

### *5.2.2 Minimizing cost versus maximizing utility*

Before presenting our coalition formation algorithms, we first present an example illustrating the dangers of attempting to minimize coalition structure cost rather than maximizing utility. There are many ways of converting utility into cost. For example, the cost of a coalition can be defined as the reciprocal of the utility or the negation of the utility [16]. We consider the former, where coalition cost is the reciprocal of the utility of the coalition and the cost of a coalition structure is the sum of the reciprocals of the utilities of the individual coalitions, which was the cost calculation method explicitly used in the Shehory and Kraus' algorithm [16].

Cost defined as the reciprocal of utility can drastically effect the relative perceived performance of coalition structures. While defining cost in such a manner has the simple effect of reversing the relative ordering of *coalitions* (i.e., if coalition $C_1$ has a higher utility than coalition $C_2$ than coalition $C_1$ will have a lower cost than coalition $C_2$), the relationship between cost and utility of *coalition structures* is more complex. As shown in the proof of Theorem 5.2, the relative ordering of coalition structures is not necessarily the reverse under cost from what it is in terms of utility. In essence, this results from the fact that a sum of reciprocals (i.e., cost of a coalition structure is the sum of the reciprocals of the utility of its

coalitions) is not the same as the reciprocal of a sum. This can lead to substantial tradeoffs between solution cost and solution utility. This is formalized in Theorem 5.2, which shows that no algorithm can simultaneously optimize, exactly or even approximately, both cost and utility.

**Theorem 5.2** *If the cost of a coalition is defined as the reciprocal of its utility, then no coalition formation algorithm can simultaneously guarantee to return a solution that is within any constant factor, or any factor that is a function of only the number of tasks and number of agents of both the lowest cost coalition structure and the highest utility coalition structure.*

*Proof* We assume that, if possible, all agents are always assigned to tasks, otherwise the minimum cost coalition structure is an empty task assignment. We also assume that agents are not assigned to tasks unnecessarily and that a coalition of agents will only be assigned to a task if they can satisfy it. These are natural assumptions, as the coalition formation problem is concerned with task completion.

Recall that the performance ratio of an algorithm $A$ on a given minimization problem instance is the ratio of the value (e.g., cost) of the solution returned by $A$ to that of the optimal solution, while in maximization problems the performance ratio of $A$ is the ratio of the value of an optimal solution to the value of the solution returned by $A$. If $A$ guarantees a performance ratio of $p$ for some function $p$ that depends only on the size of the problem instance (e.g., the number of agents, tasks and services or resources) then the value of the solution returned by $A$ is always within a factor of $p$ from the optimal (i.e., for minimization problems it is always less than or equal to $p$ times the value of an optimal solution and for maximization problems it is always greater than or equal to $\frac{1}{p}$ times the value of an optimal solution). In both maximization and minimization problems, performance ratios closer to 1 are desired, as they guarantee higher quality solutions.

Consider the following situation. There are three tasks $t_1$, $t_2$ and $t_3$. Task $t_1$ requires two agents to both perform service $s_1$ (equivalently two units of resource $r_1$). Task $t_2$ requires one agent to perform service $s_2$ (one unit of resource $r_2$). Task $t_3$ requires one agent to perform service $s_3$ (one unit of resource $r_3$). There are two agents $a_1, a_2$. Agent $a_1$ can perform services $s_1$ and $s_2$ (has one unit of resource $r_1$ and one unit of $r_2$) and agent $a_2$ can perform services $s_1$ and $s_3$ (has one unit of resource $r_1$ and one unit of $r_3$).

Assume that some coalition formation algorithm $A$ guarantees a performance ratio $p_{cost}$ in terms of the cost of the solution it generates and a performance ratio $p_{utility}$ in terms of utility. In general, both $p_{cost}$ and $p_{utility}$ can be dependent on the size of the problem (i.e., number of agents, tasks and services or resources); however, for the stated problem instance, both $p_{cost}$ and $p_{utility}$ are constant.

Let $q = max\{p_{cost}, p_{utility}\} + 1$ and let tasks $t_1, t_2, t_3$ have utilities $q$, 1, $q^2 - 1$, respectively. There are two possible coalition structures that these two agents can form. Either both agents can be assigned to task $t_1$ to form coalition structure $CS_1$, or agent $a_1$ can be assigned to task $t_2$ and agent $a_2$ can be assigned to task $t_3$ to form coalition structure $CS_2$. The utility, $u_1$, of $CS_1$ is $q$ and the utility, $u_2$ of $CS_2$ is $1 + q^2 - 1 = q^2$. However, the cost, $c_1$, of $CS_1$ is $\frac{1}{q}$ and the cost, $c_2$, of $CS_2$ is $\frac{1}{1} + \frac{1}{q^2-1} > 1$. In terms of cost, $CS_1$ is the better solution; however, in terms of utility $CS_2$ is the better solution.

$A$ must either return $CS_1$ or $CS_2$ as its solution to this problem instance. If $A$ returns coalition structure $CS_1$, then the performance ratio of $A$ in terms of total utility is

$$\frac{u_2}{u_1} = \frac{q^2}{q} = q.$$

$A$ has thus returned a solution that is a factor of $q$ off from the optimal in terms of utility. Likewise, if $A$ returns coalition structure $CS_2$ as its solution, then the performance ratio of $A$ in terms of total cost is

$$\frac{c_2}{c_1} = \frac{\frac{q^2}{q^2-1}}{\frac{1}{q}} > q.$$

$A$ has thus returned a solution that is more than a factor of $q$ off from the optimal in terms of cost.

Since $q = max\{p_{cost}, p_{utility}\} + 1$, no matter which solution $A$ returns, one of $A$'s performance guarantees will be violated.

Consider running Shehory and Kraus' algorithm [16] on the parameterized problem instance given in the proof of Theorem 2. If we set the maximum allowed coalition size to 2 then all possible coalitions will be considered as there are only 2 agents. Since Shehory and Kraus' algorithm is guaranteed to generate a solution with total cost within a constant factor of the lowest cost coalition (i.e., a constant $p_{cost}$), then by the proof of Theorem 5.2 their algorithm cannot guarantee a solution that is within any constant factor of the optimal in terms of utility.

As stated previously, there are many other methods for defining the cost of a coalition in terms of its value or utility. Another natural method for defining coalition cost is the negation of the utility. Under this definition, it is clear that a solution with total cost within a factor $r$ of the lowest cost coalition structure will also have a total utility within a factor of $r$ from the highest utility coalition structure (which in this case would simply be the lowest cost solution). However, under this situation Theorem 5.1 precludes a performance ratio, which grows logarithmically with the size of the largest allowed coalition, as all performance factors must grow almost linearly with the size of the largest allowed coalition, unless $P = NP$.

### *5.2.3 Greedy approximate coalition formation*

Both of our algorithms proceed by greedily selecting the satisfiable task (i.e., there are at most $k$ agents among the unassigned agents that can perform the task) with the greatest utility. We present two different implementations of this approach, one for the service model and one for the resource model. Our algorithm for the resource model is nearly identical to Shehory and Kraus' algorithm [16] and runs in time $O(n^k m)$, for fixed $k$, while our algorithm for the service model is based upon bipartite matching and runs in time $O(n^{\frac{3}{2}} m)$, for all fixed $k$. Our algorithm for the service model is, thus, a vast improvement over the extensions to Shehory and Kraus' algorithm made by Vig and Adams [19] for use in multi-robot domains. Vig and Adams' algorithm also ran in time $O(n^k m)$), and their algorithm's complexity was exponential in $k$. Our algorithm's asymptotic complexity is independent of $k$.

The first, preliminary, stage of our algorithm is identical to the preliminary stage of Shehory and Kraus' algorithm (Algorithm 2.1) and as such is not repeated here. Pseudo code for the second stage is provided in Algorithm 5.2. Again, note that this stage is identical to the iterative process of Shehory and Kraus' algorithm except that coalitions are selected based upon their utility rather than their weight.

Algorithm 5.3 provides psuedo code for our service model algorithm. Our approach for the service model is a centralized algorithm; however, it has a low enough computational complexity to be computed by a single agent (or by all agents simultaneously). As in the

**Algorithm 5.2** Coalition Formation Algorithm for the Resource Model

1: **while** $\mathcal{T} \neq \emptyset$ and $L_i \neq \emptyset$ **do**
2: Find the coalition $C_j \in L_i$ with the greatest utility
3: Inform all agents of the coalition weight found
4: Select the coalition and task pair, $C_{high}$ and $t_{high}$, of the highest broadcasted utility
5: $\mathcal{T} \leftarrow \mathcal{T} - \{t_{high}\}$
6: $L_i \leftarrow L_i - C_{high}$
7: Recalculate the coalition utilities of coalitions in $L_i$ that require recalculation
8: **end while**

**Algorithm 5.3** Coalition Formation Algorithm for the Service Model

1: **while** $\mathcal{T} \neq \emptyset$ **do**
2: Let $t$ be the task with the largest utility in $\mathcal{T}$
3: **if** $t$ is satisfiable by some coalition $C$ of at most $k$ agents from $\mathcal{A}$ **then**
4: Add the coalition task park $C, t$ to the current coalition structure $CS$
5: $\mathcal{T} \leftarrow \mathcal{T} - \{t\}$
6: $\mathcal{A} \leftarrow \mathcal{A} - C$
7: **else**
8: $\mathcal{T} \leftarrow \mathcal{T} - \{t\}$
9: **end if**
10: **end while**

resource model algorithm, our service model algorithm greedily selects the satisfiable task with the greatest utility.

The computationally expensive portion of Algorithm 5.3 is finding a coalition $C$ that satisfies the task $t$. A brute force search of all coalitions of size $k$ or less requires searching through $O(n^k)$ coalitions. However, as noted in the proof of Theorem 4.1, we can find a coalition of agents capable of completing $t$ using a bipartite matching algorithm. Using the algorithm of Hopcroft and Karp [9], since the number of vertices and the number of edges in the reduction described in Theorem 4.1 are both $O(n)$ (as $k$ is a constant), a maximum matching can be found in $O(n^{\frac{3}{2}})$ time. Since the bipartite matching algorithm must run a total of $m$ times, the total complexity of our algorithm is $O(n^{\frac{3}{2}}m)$.

Both Algorithms 5.2 and 5.3 provide a guarantee on the utility of the generated coalition structure:

**Theorem 5.3** *When Algorithms 5.2 or 5.3 have terminated the generated coalition structure is within a factor of* $k + 1$ *from the optimal.*

*Proof* Let $CS_{opt}$ be an optimal coalition structure and let $CS$ be a coalition structure generated by Algorithm 5.2 (Algorithm 5.3). Consider the first coalition task pair, $(C_1, t_1)$, added to $CS$ by Algorithm 5.2 (Algorithm 5.3). Since $C$ consists of at most $k$ agents and since $CS_{opt}$ consists of pairwise disjoint sets of agents, $C_1$ has a nonempty intersection with at most $k$ coalitions in $CS_{opt}$ (because if $C_1$ and $C \in CS_{opt}$ have a nonempty intersection they must share an agent in common and thus all other coalitions in $CS_{opt}$ do not contain that agent). Those $k$ coalitions cannot be in the coalition structure, $CS$, returned by Algorithm 5.2 (Algorithm 5.3). Since $t_1$ was a task with the greatest utility, $u_1$, the total utility of the sets in $CS_{opt}$ that have a nonempty intersection with $C_1$ is at most $k \cdot u_1$. There is also at most one coalition $C_{t_1}$ in $CS_{opt}$ assigned to complete task $t_1$. Since $t_1$ is assigned a potentially different coalition, the assignment of coalition $C_1$ to task $t_1$ prevents $C_{t_1}$ from being assigned to $t_1$. Thus, the total utility of the coalition task pairs in $CS_{opt}$, which assigning coalition $C_1$ to task $t_1$ invalidates, is no greater than $(k + 1) \cdot u_1$.

Likewise each coalition task pair $(C_j, t_j)$ invalidates at most an additional $k+1$ coalition task pairs in $CS_{opt}$ on top of those invalidated by the previous $j-1$ coalition task pairs added to $CS$. The set of additional coalitions invalidated by assigning $C_j$ to $t_j$ has total utility at most $(k+1) \cdot u_j$, since at each point the highest utility coalition is greedily selected. The total utility of $CS_{opt}$ is at most:

$$(k+1) \cdot u_1 + (k+1) \cdot u_2 + \cdots + (k+1) \cdot u_{|CS|}.$$

Dividing by the total utility of $CS$ we get:

$$\frac{(k+1) \cdot (u_1 + u_2 + \cdots + u_{|CS|})}{u_1 + u_2 + \cdots + u_{|CS|}} = k+1,$$

which is the stated performance ratio.

Algorithms 5.2 and 5.3, thus, have performance ratios that grow linearly with the size of the largest allowed coalition. From Theorem 5.1, we know that the performance ratio must grow at least as fast as $\Omega(k/\log k)$. Thus, Algorithms 5.2 and 5.3 have performance ratios that are nearly asymptotically optimal. We now show that the performance ratio bound presented in Theorem 5.3 is almost tight.

**Theorem 5.4** *For every $\epsilon > 0$ there is an instance of the coalition formation problem, for both the resource and service models, on which Algorithms 5.2 and 5.3 return a solution with utility exactly a factor $k - \epsilon$ from the optimal.*

*Proof* Define an instance of the coalition formation problem as follows. Let task $t$ have utility $u_t = 1$ and require one agent to perform each of the services $s_1, \ldots, s_k$. Define tasks $t_i, i = 1, \ldots, k$ to each have utility $1 - \frac{\epsilon}{k}$ and define each task $t_i$ to require only service $s_i$. Define $k$ agents, $a_1, \ldots, a_k$, such that each agent $a_i$ can perform only service $s_i$. Since our algorithms greedily satisfy the largest utility task first, task $t$ will be assigned all agents $a_1, \ldots, a_k$ since it has the greatest utility and requires a single agent to perform each service $s_1, \ldots, s_k$. Satisfying $t$ leaves the remaining tasks unsatisfiable. The total utility of this coalition structure is 1. It is easy to see that the optimal strategy is to assign each agent $a_i$ to task $t_i$ resulting in a total utility of $k(1 - \frac{\epsilon}{k}) = k - \epsilon$. The performance ratio of our algorithms in this case is $k - \epsilon$.

While we cannot hope to have a performance ratio that grows much less than linearly, additional local improvements can be made to the solutions returned by Algorithms 5.2 and 5.3 in order to decrease the performance ratio by a constant factor. There are known algorithms for the weighted $k$-set packing and the weighted $(k+1)$-claw free independent set problems that achieve better performance ratios then the $k+1$ bound presented by our algorithms [3,7]. These algorithms can reduce the performance ratio to roughly $\frac{k+1}{2}$; however, they have significant computational overhead ($O(n^{2k}m)$ or greater) and, as such, are not as appropriate for many multi-agent and multi-robot domains.

When coalition size is not restricted, our algorithm for the service model runs in $O(n^{\frac{5}{2}}m)$ time (as there are $O(n)$ vertices and $O(n^2)$ edges in the reduction to bipartite matching as $k \leq n$) and guarantees a performance ratio of $n+1$. Our algorithm, thus, guarantees a bound on the solution quality in polynomial time. This is in contrast to the results of Sandholm et al. [14] who show for characteristic function games that in order to obtain such a bound, $2^{n-1}$ coalition structures must be examined. We are able to break this bound due to the additional structure imposed on the problem under the service model. Further, our algorithm, in the unrestricted ($k = n$) situation, allows a dynamic bound to be placed at runtime on the quality

of the solution generated. From the proof of Theorem 5.3 it is easy to see that if all of the coalitions in the solution generated by our algorithm have $i$ or fewer agents, then the total utility of the generated coalition structure is at most off by a factor of $i + 1$ from the optimal. Thus, if no task requires more than $i$ services, then, in the unrestricted setting our algorithm will return a solution within a factor of $i + 1$ of the optimal.

## 6 Empirical results

We empirically validate centralized versions of our approximation algorithms for generating non-overlapping coalitions in situations with bounded coalition size by demonstrating their effectiveness on random coalition formation problem instances. For both the resource and service models, we compare the total utility of the coalition structures generated by our algorithms against the utility of an optimal solution on random problem instances.

Each resource model problem instance consisted of three random resources and a random set of tasks and agents. Each task required a random number of resources and for each required resource, a required quantity was selected randomly from the interval [1, 6]. Each task was assigned a random utility from the interval [0.1, 20.1]. A similar random process generated the set of resources possessed by each agent. Specifically, each agent possessed a random number of resources and for each resource possessed by the agent, the quantity of that resource was selected randomly from the interval [0.1, 5.1].

For the service model, there were 3 generated random services and a random set of tasks and agents in each problem instance. Each task required a random number, between 1 and 3, of services, including multiplicities. Each task was assigned a random utility from the interval [0.1, 20.1]. Each agent was able to perform a random subset of the services in the system. For both the resource and service models and in each run, the maximum allowed coalition size was 3.

We are interested in the relative performance of our algorithms with that of an optimal algorithm. For both algorithms we compute the performance ratio, the ratio of the utility of the solution returned by our algorithms to the utility of an optimal solution, on each problem instance. For both Algorithms 5.2 and 5.3 and for each number of agents and tasks, we compute both the average and the minimum performance ratio obtained on 25 random problem instances.

Figure 1 shows the average and minimum performance ratios obtained by Algorithm 5.2 on a set of random problem instances for the resource model. Similarly, Fig. 2 shows the average and minimum performance ratios obtained by Algorithm 5.3 on a set of random problem instances for the service model. In both cases, we find that the average performance ratio is always greater than 0.9. Both our algorithms on average, generate coalition structures with total utility greater than 90% of the optimal, which is much greater than the theoretical worst case. Moreover, even in the worst case, the minimum performance ratio obtained by both algorithms was greater than 0.5, or 50% of the optimal. Even the minimum performance ratios observed were significantly better, by more than a factor of 2, than the theoretical worst case performance ratio of 0.25 for this situation.

We also compare the percentage of runs on which each algorithm returns a high quality solution. Figure 3 shows the percentage of random resource model problem instances on which Algorithm 5.2 generated both an optimal solution as well as the percentage of problem instances on which it generated a solution with total utility of at least 80% of the optimal. Figure 4 shows the same set of results for Algorithm 5.3 on random service model instances. We find that while the percentage of runs on which each algorithm returns an optimal solution

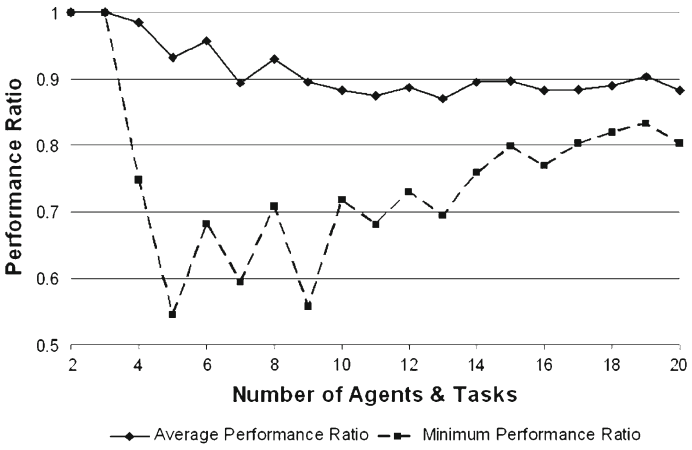


**Fig. 1** Average and minimum performance ratio of Algorithm 5.2 on random resource model problem instances.

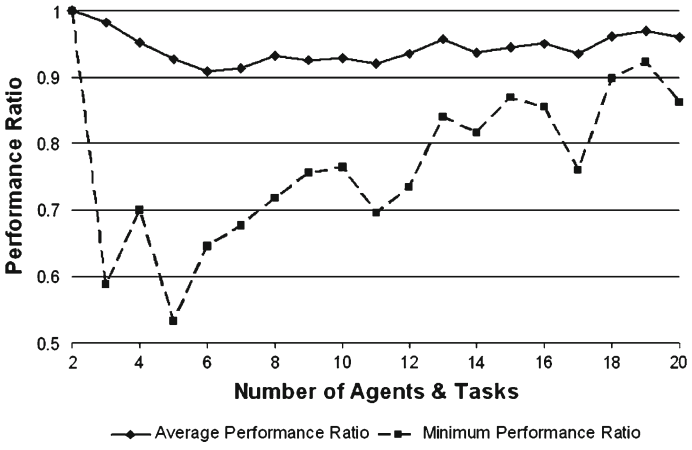


**Fig. 2** Average and minimum performance ratio of Algorithm 5.3 on random service model problem instances.

quickly decreases for larger problem sizes, our algorithms consistently, more than 75% of the time, return solutions with total utility at least 80% of the optimal.

Figure 5 shows the average and minimum performance ratios of Algorithm 5.2 on random problem instances consisting of 15 agents and tasks for different maximum allowed coalition sizes. As the graph shows, for the problem instances employed, different maximum coalition sizes did not significantly affect the resulting performance ratio of our algorithm.

To test the scalability of our algorithms, we ran both algorithms on larger problem sizes consisting of up to 50 agents and tasks. In all cases the running time of our algorithms was very fast, generating solutions in under a second, even for these larger problem sizes. Figure 6 shows the average run time of both our algorithms on problem instances consisting of between 20 and 50 agents and tasks. Each data point is averaged over 25 random problem instances.

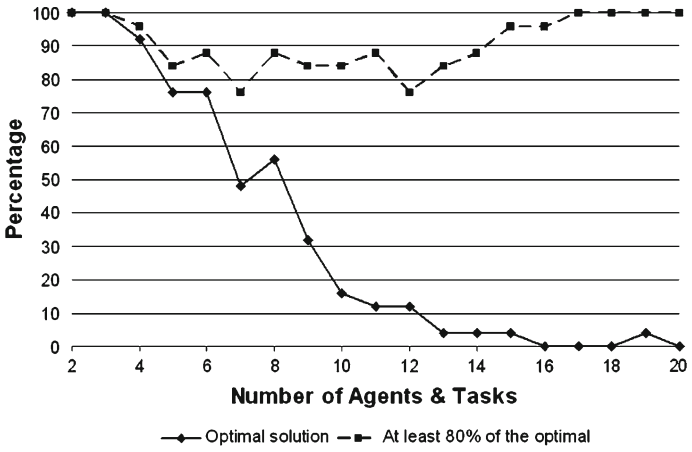


**Fig. 3** Percentage of runs on which Algorithm 5.2 generated an optimal coalition structure and a coalition structure that was 80% of the optimal on random resource model problem instances.

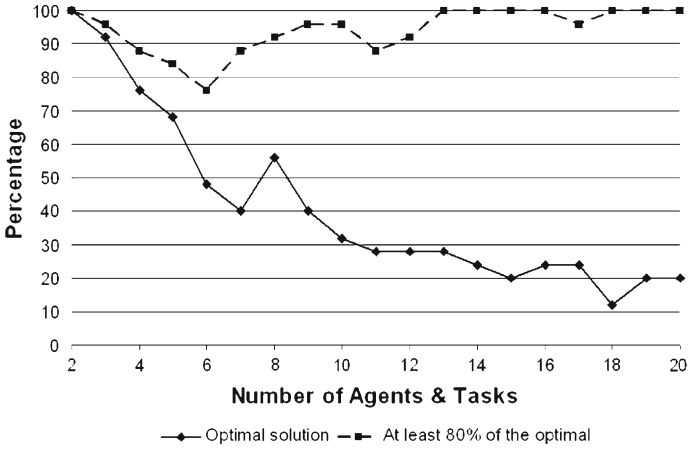


**Fig. 4** Percentage of runs on which Algorithm 5.3 generated an optimal coalition structure and a coalition structure that was 80% of the optimal on random service model problem instances.

## 7 Conclusions

This paper has considered the computational complexity of two different formalizations of the coalition formation problem. We show that this problem is NP-hard to both solve exactly and approximately within a reasonable factor for both formalizations. While the same result was known for characteristic function games [14], no result other than NP-hardness had been shown for the presented formalizations.

We considered two situations in which the coalition formation problem can be either solved exactly or approximated within polynomial time. We present a dynamic programming based algorithm for situations in which each agent is drawn from one of $j$ fixed types.

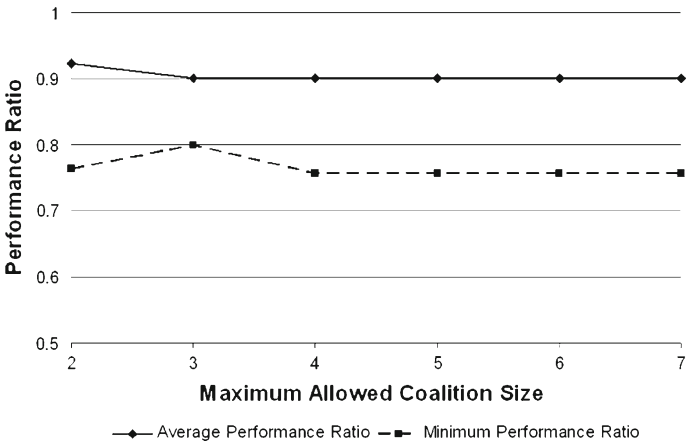


**Fig. 5** Average and minimum performance ratio of Algorithm 5.2 on random service model problem instances consisting of 15 agents and tasks for different maximum allowed coalition sizes.

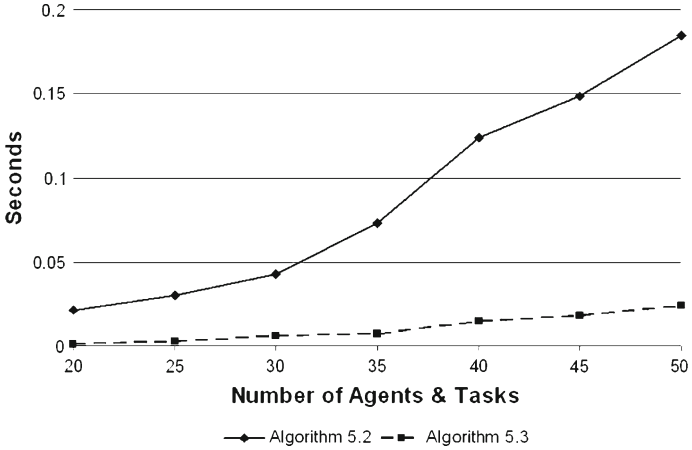


**Fig. 6** Running time in seconds of Algorithms 5.2 and 5.3 on larger problem sizes.

This algorithm is particularly applicable in the multi-robot domain, as it is unlikely that each robot will be unique.[5]

The second situation considered occurs when the size of the coalitions are bounded by a constant $k$. This situation is again of interest for multi-robot domains [20]. We present a modification Shehory and Kraus' algorithm [16] that is guaranteed to generate a coalition structure with total utility within a factor of $k + 1$ of the optimal. The bound guaranteed by Shehory and Kraus's algorithm was in terms of the cost of the coalition structure; however, as we demonstrated, minimizing total cost is not the same as maximizing total utility. Our algorithm runs in time $O(n^k m)$, which is the same as Shehory and Kraus's algorithm.

We also present an algorithm for the service model, applicable in multi-robot domains, which runs in time $O(n^{\frac{3}{2}}m)$ for any fixed $k$. Previous attempts by Vig and Adams' [19] to

[5] Assuming each robot of a particular type to be identical does not consider differences in individual robot sensors, such as wear and tear, noise, and tuning.

use Shehory and Kraus's algorithm in multi-robot domains have incurred an overhead of solving a constraint satisfaction problem on top of the $O(n^k m)$) complexity of the original algorithm. Such approaches both potentially increase the running time (although not the asymptotic complexity) as well as loose the guarantees on total solution cost of the original algorithm. Our algorithm is a vast improvement over Vig and Adams' extensions to Shehory and Kraus' algorithm, as our service model algorithm has a much lower complexity for all but the trivial case where coalitions are restricted to consist of a single agent. Further, our algorithm scales extremely well to larger sized coalitions, as its asymptotic complexity is independent of $k$, whereas Vig and Adams' algorithm is exponential in $k$.

The effectiveness of our approximation algorithms is demonstrated on random problem instances. We find that on all problem sizes tested, our algorithms generate coalition structures of average utility at least 90% of an optimal solution. This is significantly better than the theoretical worst case bound. Moreover, our algorithms ran quickly, requiring only a matter of seconds for even larger problem sizes consisting of up to 50 agents and tasks.

In contrast to the results of Sandholm et al. [14] in characteristic function games, who show that an algorithm must examine at least $2^{n-1}$ coalitions in order to guarantee any bound from the optimal. Our algorithm for the service model allows the same bound as that obtained by Sandholm et al. to be obtained in $O(n^{\frac{5}{2}}m)$, which is a large improvement over the time required to examine $2^{n-1}$ coalitions.

## References


1. Abdallah, S., & Lesser, V. (2004). Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM international conference on intelligent agent techonology* (pp. 162–168), IAT.
2. Baliyarasimhuni, S. P., & Beard, R. W. (2008). Multiple UAV coalition formation. In *American control conference* (pp. 2010–2015).
3. Berman, P. (2000). A d/2 approximation for maximum weight independent set in d-Claw free graphs. *Nordic Journal of Computing, 7*(3), 178–184.
4. Campbell, A., Wu, A. S., & Shumaker, R. (2002). Multi-agent task allocation: Learning when to say No. In *GECCO '08*: *Proceedings of the 10th annual conference on genetic and evolutionary computation* (pp. 201–208), New York, NY, USA, ACM.
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).
6. Halldórsson, M. M. (1998). Approximations of independent sets in graphs. In *APPROX '98*: *Proceedings of the international workshop on approximation algorithms for combinatorial optimization* (pp. 1–13). London, UK: Springer.
7. Halldórsson, M. M., & Chandra, B. (2001). Greedy local improvement and weighted set packing approximation. *Journal of Algorithms, 39*(2), 223–240.
8. Hazan, E., Safra, S., & Schwartz, O. (2006). On the complexity of approximating k-set packing. *Computational Complexity, 15*(1), 20–39.
9. Hopcroft, J. E., & Karp, R. M. (1973). An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal of Computing, 2*(4), 225–231.
10. Kuhn, H. (1955). The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly, 2*, 83–97.
11. Lau, H. C., & Zhang, L. (2003). Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In *ICTAI '03*: *Proceedings of the 15th IEEE international conference on tools with artificial intelligence* (p. 346). Washington, DC: IEEE Computer Society USA.
12. Rahwan, T., Ramchurn, S., Jennings, N., & Giovannucci, A. (2003). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research, 34*, 521–567.
13. Sandholm, T. (2002). An algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence, 135*(1–2), 1–54.
14. Sandholm, Y., Larson, K., Anderson, M., Shehory, O., & Tohmé, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence, 111*(1–2), 209–238.

15. Sariel, S. (2007). *An integrated planning, scheduling and execution framework for multi-robot cooperation and coordination*. PhD thesis, Istanbul Technical University.
16. Shehory, O., & Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence, 101*(1–2), 165–200.
17. Tang, F., & Parker, L. E. (2005). ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1770–1777).
18. Tosić, P., & Agha, G. (2005). Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. *Massively Multi-Agent Systems I, 3446*, 104–120.
19. Vig, L., & Adams, J. A. (2006). Multi-robot coalition formation. *IEEE Transactions on Robotics, 22*(4), 637–649.
20. Vig, L., & Adams, J. A. (2007). Coalition formation: From software agents to robots. *Journal of Intelligent Robotics Systems, 50*(1), 85–118.
21. Zuckerman, D. (2006). Linear degree extractors and the inapproximability of max clique and chromatic number. In *STOC '06*: *Proceedings of the thirty-eighth annual ACM symposium on theory of computing* (pp. 681–690). New York, NY, USA.