

# Theory and Algorithms for Coalition Formation Among Heterogeneous Agents

CS 366 - Distributed AI

Travis C. Service  
Vanderbilt University  
357 Jacobs Hall  
2201 West End  
Nashville, Tennessee 37240, USA  
tservice@acm.org

## ABSTRACT

Due to their inherent difficult and complexity, cooperation amongst autonomous agents is essential for successful completion of many real world tasks. Of interest is then how to best utilize a set of heterogenous agents to complete a set of tasks. One aspect of this 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?

This paper presents two distinct contributions to the field of multi-agent coalition formation. It is shown that unless  $P=NP$  an approximate solution to the coalition formation problem cannot be found in polynomial time to within a factor of  $O(n^{1-\epsilon})$  for any  $\epsilon > 0$ . Secondly, a comparison of two evolutionary computation based heuristic algorithms for coalition formation against a market based approach RACHNA is presented.

It is shown that, while RACHNA generates higher utility structures, the distributed evolutionary algorithm generates high quality solutions while requiring only a small fraction of the number of messages passed by RACHNA. In real-world multi-robot domains this lower communication complexity is desirable.

## 1. INTRODUCTION

Due to their inherent difficult and complexity, cooperation amongst autonomous agents is essential for successful completion of many real world tasks. Of interest is then how to best utilize a set of heterogenous agents to complete a set of tasks. One aspect of this 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? This paper presents two distinct contributions to the field of multi-agent coalition formation.

Firstly, a complexity theoretic approach is taken. It is shown that, in addition to being an NP-hard problem, the general coalition formation problem is also an APX-hard problem and not in APX<sup>1</sup>. This result shows that not only is the coalition formation problem hard to solve but it is also hard to approximate. Thus, we can not, in polynomial time, find an approximate solution which is guaranteed to

<sup>1</sup>APX is the class of optimization problems which have polynomial time algorithms guaranteed to produce a solution within some constant factor of the optimal.

be within a constant factor of the optimal solution unless  $P=NP$ . Furthermore, by the results presented in [13], we can conclude that it is not even possible in polynomial time to find an approximate solution which is within a factor of  $O(n^{1-\epsilon})$  for any  $\epsilon > 0$ , where  $n$  is the number of tasks, unless  $P = NP$ .

Secondly, two different heuristic based coalition formation algorithms based upon evolutionary computation are presented and compared against the market based approach RACHNA<sup>2</sup> [11]. RACHNA is compared against a cooperative coevolutionary based approach as well as a distributed evolutionary computation based approach. The three algorithms are compared along the following three dimensions:

- utility of the generated coalition structures,
- running time, as the experiments were run on a single machine a discussion is provided about how well each technique will distribute,
- communication complexity, in the multi-robot domain real-world issues have an affect on the ability of inter-robot communication as such techniques which produce quality solutions while minimizing communication overhead are desirable.

The empirical study presented in this paper shows that while RACHNA still outperforms the both the heuristic approaches in terms of the utility of the generated solutions, the distributed evolutionary algorithm is far superior in terms of the number of messages passed between agents. Further, the computations performed by each individual agent when using the distributed evolutionary algorithm are almost completely independent of the other agents (however, this depends upon the particular parameters used). This independence leads to almost perfect distributability of the computations amongst the set of agents.

The remainder of this paper is as follows. Section 2 provides a survey of work related to this paper. The two variants of the formal coalition generation problem considered in this paper are defined in Section 3. Section 4 provides the necessary background in complexity theory. In Section 5 it is shown that unless  $P=NP$  we find an approximate solution, in polynomial time, within a factor of  $O(n^{1-\epsilon})$ , for any

<sup>2</sup>Robot Allocation through Coalitions using Heterogeneous Negotiating Agents

$\epsilon > 0$ , to either of the two considered variants of the coalition formation problem. The two evolutionary computation based approaches are described in Section 6. A comparison of both approaches against the market based approach RACHNA is presented in Section 7. In Section 8 a discussion of our results is presented followed by some conclusions and future work in Section 9.

## 2. RELATED WORK

Much work has gone into task allocation and coalition formation in multi-agent systems [6, 10, 9].

[9] takes 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  $\nu_S$ . A coalition structure  $CS$  is then simply a partitioning of the agents. The goal is then to find the coalition structure  $CS^*$  which maximizes the sum of the coalition values:

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

They show that in order to obtain any bound on the solution quality a search algorithm is required to visit at least  $O(2^{n-1})$  coalition structures, where  $n$  is the number of agents. They provide an algorithm which examines those coalition structures which partition the agents into one or two partitions 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 potentially arbitrarily high values. In order to get within a guaranteed bound on the optimal solution, every coalition must be present in at least one of the examined coalition structures, as an unexamined coalition could potentially have a significantly high value.

An issue with using characteristic function games for modeling coalition formation for task allocation is that characteristic function games do not take into account the task which each coalition is assigned to. That is, the coalition value of a coalition  $S$  is independent of everything other than 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 not only on the members of  $S$  but also on the task  $S$  is to perform.

A classification of coalition formation problems is presented in [6]. 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 presented 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 task at simultaneously, and as such coalitions may need to be pairwise disjoint.

A heuristic based coalition formation algorithm is presented in [10]. They impose a limit on the maximum size of a coalition,  $k$ , and present an algorithm which runs in  $O(n^k m)$  time, where  $n$  is the number of agents and  $m$  is the number of tasks. They also proof a logarithmic performance bound of the cost of the returned coalition structure from the optimal, where the cost is of a coalition is defined by the reciprocal of its value. However, this 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.

An anytime algorithm for coalition formation in characteristic function games based on the branch and bound technique is presented in [7]. The search space is partitioned into disjoint subspaces and an upper and lower bound is calculated for each allowing potentially large portions of the search space to be pruned from the search.

Many of the algorithms developed for coalition formation are anytime algorithms [7, 10, 9]. Anytime algorithms permit the agents to dynamically decide if 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 produce at least a reasonable solution whereas non-anytime algorithms may not have sufficient time to terminate. The evolutionary computation based approach presented in this paper is another example of an anytime algorithm.

[11] considered the coalition formation algorithm in the context of multi-robot domains. Some of the major differences between multi-agent and multi-robot domains are presented and a modified version of the multi-agent coalition formation algorithm presented in [10] is presented for the multi-robot domain. Further, [11] points out some shortcomings of this heuristic based approach for the multi-robot domain. The market based approach RACHNA attempts to address some of those issues with coalition formation in multi-robot domains [11]. The RACHNA system leverages redundancies in agents. Each agent is modeled as being able to perform a set of services. 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.

The RACHNA system examined in this paper takes a combinatorial auction approach. However, [8] shows, via a reduction from the maximum weighted independent set problem, that, unless  $P=NP$ , it is impossible to approximate the winner of a combinatorial auction in polynomial time to within a bound of  $O(n^{1-\epsilon})$  for any  $\epsilon > 0$ <sup>3</sup>. This result; however, does not imply that approximating the optimal coalition structure within a factor of  $O(n^{1-\epsilon})$  is impossible unless  $P=NP$ . It just shows that techniques that make use of general combinatorial auctions to solve the coalition formation problem cannot achieve such bounds in polynomial time. However, in Section 5 we provide a proof that the general coalition formation problem cannot be approximated within a factor of  $O(n^{1-\epsilon})$  for any  $\epsilon > 0$  unless  $P=NP$ .

There are many problems in theoretical computer science which have similarities with the coalition formation problem. For an example, [10] discusses the relationship between both the set covering problem and set partitioning problem and the coalition formation problems.

[2] considers a different aspect of task allocation in multi-agent systems. They consider the situation where the group of agents are 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"

<sup>3</sup>The theorem presented in [8] states the inapproximability result for  $NP \neq ZPP$ ; however, this can be strengthened to  $P \neq NP$  using the results of [13].

for tasks (i.e., the difficulty of a task compared to the ability of the agent). Compared to a greedy and a heuristic, fixed tolerance, based approach they find that, under certain conditions on the distribution of task difficulties and lengths, the dynamic approach they present better 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 they test, it requires no explicit communication between agents, permitting near perfect scaling, and is robust to changes in the agents.

### 3. PROBLEM DESCRIPTION

Two different formulations of the coalition formation problem are considered in this paper. Let  $\mathcal{T}$  be a set of  $n$  tasks and  $\mathcal{A}$  be a set of  $m$  agents. The first formalization presented in Definition 3.1 is employed in the RACHNA system [11]. The second formalization, Definition 3.2, has been used previously in multi-agent domains [10]. In both formalizations it is assumed that the agents are group rational and try to maximize the total utility gained by the group.

**DEFINITION 3.1 (COALITION FORMATION I).** *Each task  $t \in \mathcal{T}$  has an associated vector  $S_t = (s_1, s_2, \dots, s_p)$  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 also a vector of services  $S_a = (s_1, s_2, \dots, s_p)$  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. Given an assignment of agents to tasks and corresponding services, a task  $t$  is satisfied if and only if every service required by that task is performed by some agent.*

**DEFINITION 3.2 (COALITION FORMATION II).** *Each task  $t \in \mathcal{T}$  has an associated vector  $R_t = (r_1, r_2, \dots, r_p)$  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 also a vector of resources  $R_a = (r_1, r_2, \dots, r_p)$  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, 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 3.1 can be seen as a modification to Definition 3.2 for use in multi-robot domains where the individual robots (agents) can not share resources. The two problem definitions differ only in that the second allows each agent to contribute to multiple of the tasks required resources while the first requires each agent to contribute to only one. This paper focuses on the first definition for the empirical comparisons of the evolutionary computation based approaches to RACHNA. However, the complexity theoretic results and algorithms apply to both definitions.

### 4. COMPLEXITY THEORY PRELIMINARIES

We assume the reader is familiar with some standard basic complexity theory background including the notions of polynomial many-one reductions and completeness as well

as the complexity classes P and NP. This section covers the basics of optimization problem approximation as well as the corresponding complexity classes and notions of problem reduction.

A deterministic algorithm is said to be a polynomial time  $c$ -approximate algorithm, for a constant  $c$ , to an optimization problem  $O$  if it both runs in polynomial time and for each problem instance  $I$  outputs an answer which is within a factor of  $c$  from the optimal solution to  $I$ . The complexity class APX consists of those optimization problems for which there exists a polynomial time  $c$  approximation algorithm for some constant  $c$ .

An optimization problem  $O$  is said to have a polynomial time approximation scheme (PTAS) if for every fixed  $\epsilon > 0$  there exists a polynomial time algorithm which is a  $(1 + \epsilon)$ -approximate algorithm for  $O$ . Thus, if an optimization problem emits a PTAS we find a solution which is within a factor of  $1 + \epsilon$  of the optimal solution, for a fixed  $\epsilon$ , in polynomial time.

A PTAS reduction from an optimization problem  $A$  to an optimization problem  $B$  is a triple  $(f, g, \alpha)$  where:

1.  $f$  is a polynomial time many-one reduction from  $A$  to  $B$ ,
2.  $g$  is a polynomial time algorithm mapping and instance  $x$  of  $A$ , an approximate solution  $f(x)$  of  $B$  and an error parameter  $\epsilon$  to an approximate solution of  $A$
3.  $\alpha$  is a polynomial time algorithm mapping error parameters for  $A$  to error parameters for  $B$ .

PTAS reductions have similar properties for approximation classes as polynomial many-one reductions have for decision problem class (e.g., transitivity).

An optimization problem  $O$  is said to be APX-hard if there is a PTAS reduction from every problem in APX to  $O$ . Unless  $P = NP$  no APX-hard optimization problem has a PTAS.

It is a consequence of the PCP theorem [1] that the maximum independent set problem and maximum clique problem are not in APX (i.e., cannot be approximating within a constant factor in polynomial time unless  $P=NP$ ). Since the PCP theorem additional results have strengthened the inapproximability of these problems. [13] shows that unless  $P=NP$  both cannot be approximated within a factor of  $O(n^{1-\epsilon})$  for any  $\epsilon > 0$ . This is an important result, since for both there is a trivial approximation algorithm which gets within a factor of  $O(n)$  by simply returning a single vertex.

### 5. HARDNESS OF APPROXIMATION

This section considers the ‘‘approximability’’ of both formalizations of the coalition formation problem. It is shown that in addition to being NP-hard optimization problems they are also APX-hard and not in APX, implying that we can not hope for a polynomial time  $c$ -approximation algorithm for any constant  $c$ .

**THEOREM 5.1.** *Both Problems I and II are APX-hard*

**PROOF.** We will construct a PTAS reduction from the APX-hard problem maximum independent set problem to the coalition formation problem I formalization and then describe how this reduction applies to problem II.

Let  $G = (V, E)$  be an arbitrary graph and let  $n$  and  $m$  denote the number of vertices and edges, respectively. For each vertex  $v_i$  we define a task  $t_i$  and for each edge  $e_i$  we 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 service  $s_i$ ). For each edge incident to a node  $v_i$  we 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  $v_i \in I$  we assign the agents corresponding to each edge incident to  $v_i$  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  $G$  consisting of each vertex  $v_i$  corresponding to a satisfied task  $t_i$ . This 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 neighboring vertices' corresponding task can be satisfied.

Thus, there is a one-to-one mapping between the independent sets of  $G$  and coalition structures such that if the independent set in  $G$  has  $k$  vertices the utility of the corresponding coalition structure is  $k$ .

The reduction above can clearly be done in polynomial time.

The mapping of approximate solutions to the coalition formation problem to approximate solutions to the independent set problem take place as described above and hold for the same error parameters.

Thus, an approximation algorithm to the coalition formation problem provides an approximation algorithm for the maximal independent set problem.  $\square$

The above reduction applies also to problem definition II. 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.

**COROLLARY 5.2.** *Coalition Formation is not approximable within a factor of  $O(n^{1-\epsilon})$ , where  $n$  is the number of tasks, for any  $\epsilon > 0$  unless  $P=NP$ .*

**PROOF.** This follows directly from the reduction in Theorem 5.1 and the results of [13].  $\square$

Note that we can trivially find an approximate solution to the coalition formation problem within a factor of  $O(n)$  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  $O(n/\text{polylog}(n))$  [4]. If so this would carry 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. For example, in the case where each agent belongs to one of  $k$  possible types<sup>4</sup>, where agents of the same type are indistinguishable from one another, we can solve the coalition formation via a dynamic programming approach in  $O(nm^{2k})$  time. Furthermore if the agents are all homogeneous, all agents are indistinguishable, we can solve the coalition formation problem via a greedy approach in  $O(n \log n)$  time (or  $O(nm)$  time with a dynamic programming approach)<sup>5</sup>.

## 6. EVOLUTIONARY COMPUTATION APPROACH

Since the coalition formation problem is NP-hard to solve exactly and also NP-hard to solve approximately within a reasonable factor, this paper turns to heuristic based approaches to solve the general coalition formation problem. Two coalition formation approaches were developed and tested based upon evolutionary computation. Both approaches are heuristic based and as such provide no guarantees on the quality of the solution returned

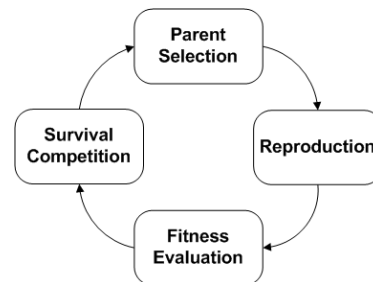
The first approach is based on cooperative coevolution [12, 5] while the second uses a distributed island based approach. Both approaches are described here.

### 6.1 Overview

Before presenting the two heuristic evolutionary based algorithms for coalition formation, a brief overview of evolutionary computation is presented. For more information the reader is directed to [3].

#### 6.1.1 Evolutionary Computation

Evolutionary algorithms are a class of stochastic population based heuristic algorithms. A population of potential solutions are evolved over time through the simulated processes of recombination and mutation. A fitness is assigned to each potential solution in the population in proportion to how well it solves the problem at hand. More fit individuals have a greater chance of reproducing as well as surviving to future generations, as less fit individuals are pruned from the population. Figure 1 shows the steps in the basic evolutionary algorithm.



**Figure 1: Basic evolutionary algorithm cycle, excluding initialization and termination.**

<sup>4</sup>This ignores differences between individual sensors on identical robots.

<sup>5</sup>The author has developed both of these algorithms; however, they are out of the scope of this paper and as such are not included.

### 6.1.2 Cooperative Coevolution

In cooperative coevolution an optimization problem is subdivided into  $n$  subproblems. Individuals in each subproblem are evolved independently and come together for joint fitness evaluation [12]. Figure 2 shows the basic coevolutionary algorithm cycle.

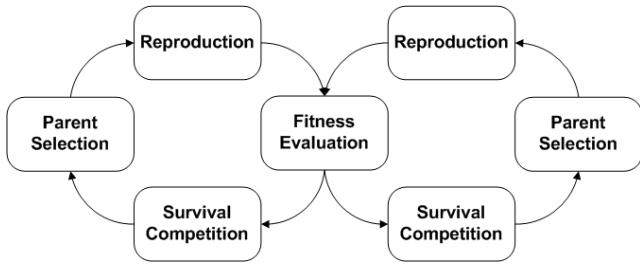


Figure 2: Basic coevolutionary algorithm cycle, excluding initialization and termination.

## 6.2 Cooperative Coevolutionary Algorithm

The first algorithm to be used is based on cooperative coevolution [12]. In cooperative coevolution an optimization problem is decomposed into  $n$  separate subproblems each of which is optimized simultaneously [12, 5]. This naturally carries over to the coalition formation problem; each agent simultaneously optimizes its assignment to a particular task. This process is naturally distributable as the individual optimizations are performed semi-independently from one another, with communication arising only during fitness computation.

Pseudo code for the coevolutionary based approach is provided in Algorithm 6.1 and Algorithm 6.2. Algorithm 6.1 presents the outer loop of the algorithm which is simply a standard coevolutionary algorithm. The algorithm takes two parameters:  $k$  which indicates the agents id or number (i.e., the  $k$ -th agent) and the population size.

---

#### Algorithm 6.1 *CoEA\_Coalition\_Formation(k, popSize)*

---

```

1: while TerminateCriteriaNotMet do
2:   if FirstGeneration then
3:     curTaskPop ← InitRandom(popSize, TASKS)
4:   else
5:     EvolveNewPopulation(curTaskPop)
6:   end if
7:   EvaluatePopulation(curTaskPop)
8: end while
9: Select best selectedTask ∈ curTaskPop

```

---

The differences between a standard cooperative coevolutionary algorithm and the distributed multi-agent based approach is in the *EvaluatePopulation* function shown in Algorithm 6.2. The primary difference being that the agents must communicate the contents of their respective populations to one another before they can evaluate the fitness (task utilities) of the members of the current population.

## 6.3 Distributed Evolutionary Algorithm

The second approach uses a distributed evolutionary algorithm to heuristically find a high quality coalition structure.

---

#### Algorithm 6.2 *EvaluatePopulation(k, curTaskPop)*

---

```

1: evalSet ← SelectEvaluationSet(curTaskPop)
2: BroadcastEvaluationSet(evalSet, k)
3: for all Robots  $i \neq k$  do
4:   evalSet $i$  ← ReceiveBroadcastEvaluationSet( $i$ )
5: end for
6: for all task ∈ curTaskPop do
7:   taskutility = 0
8: end for
9: coalitionStructures ← evalSet1 × ... × evalSet $k-1$  ×
  curTaskPop × evalSet $k+1$  × ... × evalSet $n$ 
10: for all allocation ∈ coalitionStructures do
11:   task ← allocation( $k$ )
12:   taskutility ← taskutility + Evaluate(allocation)
13: end for

```

---

Algorithm 6.3 provides pseudo code for the distributed evolutionary algorithm approach.

---

#### Algorithm 6.3 *DistributedEA()*

---

```

1: while TerminateCriteriaNotMet do
2:   if FirstGeneration then
3:     curPop ← InitRandom(popSize, TASKS)
4:   else
5:     EvolveNewPopulation(curPop)
6:   end if
7:   for all assignment ∈ curPop do
8:     assignmentutility ← Evaluate(assignment)
9:   end for
10:  with 30% probability broadcast best assignment to
  every agent
11: end while
12: Select best selectedTask ∈ curTaskPop

```

---

The agents probabilistically share assignments through broadcasted messages. In the evolutionary computation field this model is referred to as an island model. Each island is evolved separately and occasionally individuals cross over between islands [3].

The agents share individuals by broadcasting their highest fitness individual to each other after every generation with a certain probability. Upon receipt of a broadcasted assignment, each agent adds that individual to its current population, with a certain probability.

## 7. EMPIRICAL COMPARISON

Both the cooperative coevolutionary and the distributed evolutionary algorithms were compared against the market based approach RACHNA [11].

A minimum bid increment of 0.8 was used with RACHNA. Table 1 provides the parameters used with the distributed evolutionary algorithm. Only mutation was used for the cooperative coevolutionary based approach and a population of size 50 was evolved for 400 generations.

Modeling each agent's task assignment as a separate subproblem requires  $m$  population, where  $m$  is the number of agents. Using full mixing (i.e., each agent's potential assignments evaluated against the potential assignments being considered by all other agents) results in an exponential, in  $m$ , number of evaluations required per agent. Specifically, for a population size of  $p$ , under full mixing, each agent

must compute the value of  $p^m$  coalition structures. In order to make the coevolutionary approach tractable, each agent will broadcast a single of its currently considered potential task assignments to all other agents to use for fitness evaluation. This results in each agent evaluating only  $p$  coalition structures; however, it also decreases the accuracy of the fitness evaluations.

Each of the data points presented are averaged over 25 runs of each algorithm on randomly generated problem instances consisting of the stated number of agents and tasks 10 services. Each agent is randomly assigned a set of between 1 and 10 possible services it can provide. Each task requires a random set of between 1 and 10 services. The task utilities are chosen uniformly from  $[0.1, 20.1]$ . Thus, the task utilities are within a range of 20 possible values and no task is assigned a utility of 0.

**Table 1: Distributed Evolutionary Algorithm Parameters**

Parameter	Value
parent & survival selection	binary tournament selection
recombination	binary crossover
mutation	random assignment of a particular robot to a new service
crossover probability	80%
mutation probability	30%
population size	50
number of generations	400

## 7.1 Utility

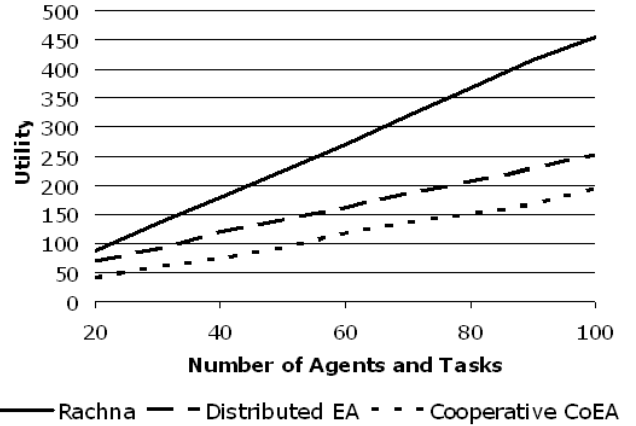
RACHNA consistently produced higher quality coalition structures, followed by the distributed evolutionary algorithm and then the cooperative coevolutionary algorithm. Figure 3 shows the average utility of the coalition structures generated by each algorithm. As can be seen RACHNA consistently finds higher utility coalition structures than the two evolutionary based techniques. The utility of the coalition structures generated by all three techniques improved linearly with the size of the problem.

It is interesting to note that the utility of the solution returned by RACHNA seems to improve faster, as the problem size grows, than the utility of the evolutionary based techniques do. Also, the rate at which the utility improves for the two evolutionary based techniques appears to be the same (i.e., same slopes); however, the distributed evolutionary algorithm provided higher quality results.

## 7.2 Running Time

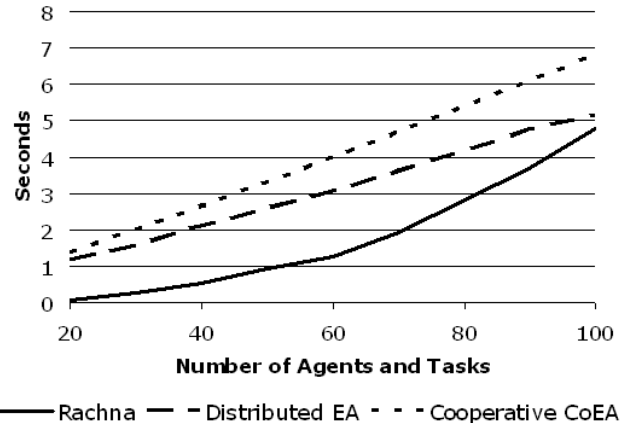
The running times of each algorithm for different problem sizes was measured. Since all agents were run on a single machine it is difficult to measure the total computation time of the distributed versions of the algorithms; however, based upon the workings of each techniques we can hypothesize.

In the ideal case if we distribute a computation that requires time  $f(n)$  to compute on a single agent, for a problem of size  $n$ , to  $m$  agents, our distributed version would require  $\frac{f(n)}{m}$  (i.e., a speed up of a factor of  $m$ ). As a market based approach, RACHNA requires a fair amount of synchronization between the bidders and the auctioneers and as such



**Figure 3: Utility of the coalition structures generated by each algorithm.**

we should not expect this ideal situation. However, with the evolutionary computation based approaches, since most of the computations performed by each agent are independent of one another, we should expect a speed up of nearly a factor of  $m$ .



**Figure 4: Estimated average distributed running time of each algorithm.**

## 7.3 Communication

While the distributed evolutionary algorithm produced coalition structures of lesser quality than those produced by RACHNA, the distributed evolutionary algorithm used an order of magnitude fewer messages than RACHNA did. In multi-robot domains where communication issues are to be expected, a reduction in the total number of messages required is desirable. Note that, the below graph assumes that there are no broadcast messages. Thus, in order to send a message to each of the other  $m - 1$  agents, the message must be sent  $m - 1$  times. As all of the messages sent by both evolutionary based techniques, allowing broadcast messages will reduce the total number of messages by a factor of about  $m$ . While the amount of messages sent

by RACHNA will be scaled down similarly, not all messages used by RACHNA are meant to be broadcasts and as such the number of messages sent is expected to reduce by less than a factor of  $m$ . Figure 5 shows the number of messages used by both RACHNA and the distributed evolutionary algorithm. The messages used by the cooperative coevolutionary algorithm was significantly more than used by the other two and is not included in the graph.

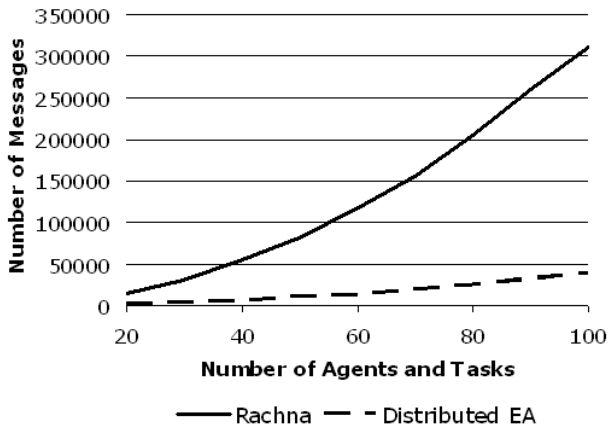


Figure 5: Average number of messages used by RACHNA and the distributed evolutionary algorithm.

The number of messages used can also potentially have a large impact on the running time of the algorithms. For example, in cases where the agents all share a common medium, agents may be required to wait for their turn to use the channel.

## 8. DISCUSSION

The results of this paper bring up several points of discussion.

### 8.1 Implications of Hardness of Approximation

The complexity theoretic result presented in Section 5 shows that not only is the general coalition formation problem NP-hard to solve exactly but it is also NP-hard to approximate within a factor of  $O(n^{1-\epsilon})$ . Such results indicate that, unless  $P=NP$ , we cannot even find a solution with a guarantee of being within a non-trivial factor of the optimal solution in polynomial time. Thus, in order to generate an approximate, or optimal, solution, we must either do so with no guarantee of running in polynomial time, or restrict our attention to tractable, but still applicable, restricted classes.

A few such restricted classes were described in Section 5. For any arbitrary but fixed  $k$ , if each agent falls is one of among  $k$  types then we can solve the coalition formation problem in  $O(nm^{2k})$  time. Further, for  $k = 1$  (i.e., all agents are homogeneous) we can solve generate an optimal coalition structure in  $O(n \log n)$  time via a greedy algorithm. Such an approach (i.e., considering a set of agents draw from a fixed set of  $k$  types) seems particularly appropriate in the multi-robot domain. It seems unlikely that each robot would be of a distinct type. However, such a model ignores differ-

ences between individual robot sensors and actuators. Future work is needed to determine if this simplification is appropriate in real-world domains.

While this paper has proven the hardness of approximating two different formalizations of the coalition formation problem, it is expected that a similar reduction will be applicable in most other formalizations. Specifically, any formalization which permits tasks to require specific agents in order to be satisfied should allow for a similar reduction from the independent set problem.

One can also imagine situations where each task requires a certain amount of time to complete (possible a function of the capabilities of the agents assigned to complete it) and may be ordered by a precedence relation and the goal is to complete all tasks as quickly as possible. A situation similar to this was considered in [2]. This scenario is essentially a scheduling problem and similar hardness of approximation can be proven using a reduction from the graph coloring problem.

### 8.2 Distributing RACHNA

The experiments performed using RACHNA in [11] used a centralized version of RACHNA. In this work a distributed version of RACHNA was created to test the algorithm in a more realistic setting. In order to make the distributed version of RACHNA to work a few modifications were made to the algorithm. These include:

- after making a bid, the task agents assume that they have been awarded the requested services until otherwise informed,
- after accepting a bid, the service agent broadcasts the current robot salaries and assignments of robots capable of performing its service.

These changes were relatively minor and did not appear to change the effectiveness of the algorithm. Thus, RACHNA can successfully be used in a distributed multi-agent setting, as well as in the standalone centrally computed situation as has been previously the case [11].

### 8.3 Communication Overhead

Even though RACHNA outperformed the distributed evolutionary algorithm in terms of the utility of the generated coalition structure, the distributed evolutionary algorithm, with the parameter set used, required an order of magnitude fewer messages to be passed. In real-world settings where communication issues could arise this difference could be significant. For example, if all of the agents are communicating over a common medium then agents may be forced to wait for the channel to be free before sending messages. This waiting could result in additional time required to run the algorithm. Additional tests need to be done to determine the effect of large numbers of messages on algorithm run time in real world settings.

An added benefit of the distributed evolutionary algorithm is that the number of messages required can be reduced to  $O(m)$  if individual assignments are not shared between agents during evolution (i.e., no island model). In that case the only communication required is at the very end when an actual coalition structure must be agreed upon by the agents. However, this essentially turns the distributed coalition structure generation into each  $m$  agent generating

a coalition structure on its own and the group selecting the best.

If choosing between RACHNA and a distributed evolutionary computation approach, considerations of communication complexity and solution quality need be considered and a trade-off made. Often, the determination of the appropriate algorithm will be domain specific (e.g., domains where communication is costly might favor the evolutionary computation approach).

## 8.4 Project Process

Throughout the course of this project no major hurdles were encountered. The schedule was reorganized a couple times to include the addition of new algorithms (i.e., the distributed evolutionary algorithm and the player stage version of RACHNA).

## 9. CONCLUSIONS & FUTURE WORK

This paper presented a empirical comparison of two coalition formation algorithms based on evolutionary computation against the market based approach RACHNA [11]. While RACHNA outperformed both the evolutionary computation based techniques in terms of the total utility of the generated coalition structures, the distributed evolutionary computation approach has the advantage in that it uses an order of magnitude fewer messages. Additionally, the number of messages required by the distributed evolutionary computation approach can be further reduced to  $O(m)$ , where  $m$  is the number of agents, with the appropriate parameters.

The cooperative coevolutionary based approach was found not to work well for coalition formation as employed. With each agent coevolving its own task assignment, the algorithm could not scale up to large problem sizes. It also underperformed the other two algorithms in terms of both the number of messages required and the utility of the generated coalition structures.

This paper also presented a proof demonstrating that not only is the general coalition formation problem NP-hard to solve exactly, but it is also NP-hard to approximate to within a reasonable factor. Further, it is argued that, while there are other possible formalizations of the coalition formation problem, it is expected that all similar formalizations will also be hard to approximate and sufficient conditions are provided for hardness of approximation. However, there are some situations for which the analysis presented in this paper does not directly apply. Future work is required to identify and classify such situations.

## 10. REFERENCES

- [1] S. Arora and S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. In *SFCS '92: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 2–13, Washington, DC, USA, 1992. IEEE Computer Society.
- [2] Adam Campbell, Annie S. Wu, and Randall Shumaker. Multi-Agent Task Allocation: Learning When to Say No. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 201–208, New York, NY, USA, 2008. ACM.
- [3] Kenneth De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, 2006.
- [4] Magnús M. Halldórsson. Approximations of Independent Sets in Graphs. In *APPROX '98: Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 1–13, London, UK, 1998. Springer-Verlag.
- [5] Daniel Hillis. Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure. *Physica D Nonlinear Phenomena*, 42:228–234, June 1990.
- [6] Hoong Chuin Lau and Lei Zhang. 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*, page 346, Washington, DC, USA, 2003. IEEE Computer Society.
- [7] Talal Rahwan, Sarvapali Ramchurn, Nicholas Jennings, and Andrea Giovannucci. An Anytime Algorithm for Optimal Coalition Structure Generation. *Journal of Artificial Intelligence Research*, 34:521–567, 2009.
- [8] Tuomas Sandholm. An Algorithm for Optimal Winner Determination in Combinatorial Auctions. In *Artificial Intelligence*, pages 542–547, 1999.
- [9] Tuomas Sandholm, Kate Larson, Martin Anderson, Onn Shehory, and Fernando Tohmé. Coalition Structure Generation with Worst Case Guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.
- [10] Onn Shehory and Sarit Kraus. Methods for Task Allocation via Agent Coalition Formation. *Artificial Intelligence*, 101(1-2):165–200, May 1998.
- [11] Lovekesh Vig and Julie A. Adams. Coalition Formation: From Software Agents to Robots. *Journal of Intelligent Robotics Systems*, 50(1):85–118, 2007.
- [12] R. Paul Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 2003.
- [13] David Zuckerman. 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*, pages 681–690, New York, NY, USA, 2006. ACM.