A Framework for Multi-Robot Coalition Formation

Lovekesh Vig and Julie A. Adams

Electrical Engineering and Computer Science Department,
Vanderbilt University,
Nashville,
TN 37212
{lovekesh.vig, julie.a.adams}@vanderbilt.edu

Abstract. Task allocation is a fundamental problem that any multi-robot system must address. Numerous multi-robot task allocation schemes have been proposed over the past decade. A vast majority of these schemes address the problem of assigning a single robot to each task. However as the complexity of multi-robot tasks increases, often situations arise where multiple robot teams need to be assigned to a set of tasks. This problem, also known as the coalition formation problem has received relatively little attention in the multi-robot community. This paper provides a generic, task independent framework for solutions to this problem for a variety task environments. In particular, the paper introduces RACHNA, a novel auction based coalition formation system for dynamic task environments. This is an extension to our previous work which proposed a static multi-robot coalition formation algorithm based on a popular heuristic from the Distributed Artificial Intelligence literature.

1 Introduction

Task allocation is a problem that must be addressed by any multi-robot system. Over the past decade, numerous solutions to the Multi-Robot Task Allocation (MRTA) problem have been proposed in the literature. However, most of the proposed solutions assign individual robots to individual tasks. In other words, the existing solutions operate under the assumption that each task may be performed by a single robot, also called the single-task single-robot (ST-SR) assignment problem [1].

As multi-robot systems evolve, multi-robot tasks are becoming more complex. The increase in task complexity results in situations where a task cannot be completed by a single robot and it becomes necessary to assign a team of robots to an individual task. This problem is called the single-task multiple-robot (ST-MR) allocation problem and is the central theme of this work. In this paper the problem of allocating tasks to disjoint robot teams is investigated. Each such team is referred to as a coalition. The set of all disjoint coalitions is called the coalition structure. Finding the optimal coalition structure is called the coalition

formation problem and has been proven to be NP-complete [2]. Solutions to the coalition formation problem have many potential applications, especially in situations where tasks are located at considerable distances from one another and teams of robots need to be dispatched to different locations to autonomously complete their designated tasks.

Gerkey and Matarić [3] point out that despite the existence of various multi-agent coalition formation algorithms, none of these algorithms had been demonstrated in the multi-robot domain. Although not heavily pursued in Multi-Robot Systems (MRS), coalition formation is an established area of Distributed Artificial Intelligence (DAI) research, and various heuristics have been proposed that yield good, tractable, sub-optimal solutions. However, these solutions make underlying assumptions that are not applicable to the multiple-robot domain, hence the existence of a discrepancy between the multi-agent and multi-robot coalition formation literature. Our prior work [4] identifies these assumptions and provides modifications to multi-agent coalition formation algorithms to enable their use in the multi-robot domain. In this paper we outline extensions of this work so as to provide a framework for allocation of multi-robot (MR) tasks. In particular, we present RACHNA[1], a novel dynamic, market-based architecture for allocating multi-robot tasks based on individual robot capabilities.

The coalition formation approaches discussed in this paper are completely task independent, i.e. task allocation is performed without making any underlying assumptions about the types of tasks. Also no assumption is made regarding the approach used to perform the task, i.e. the underlying coordination mechanism could be behavior based, swarm based or economy based without impacting team formation.

The remainder of the paper is organized as follows. Section 2 provides the related work. Section 3 identifies some important differences between software agent environments and multi-robot environments. Section 4 provides a brief glance of our previous work which lead to the ideas presented in this paper. Section 5 formulates the coalition formation problem as an instance of the matching problem. Section 6 outlines the different kinds of task environments and introduces RACHNA, a novel, market-based, dynamic coalition formation system. Section 7 discusses the conclusions and identifies areas for future work.

2 Related Work

A number of elegant solutions to the task allocation problem have been proposed in the literature. The ALLIANCE [5] architecture uses motivational behaviors to monitor task progress and dynamically reallocate tasks. ALLIANCE employs a variant of the subsumption architecture [6] and makes use of "behavior sets" to enable a robot to perform versatile tasks. Recently Low et al. [7] proposed a swarm based approach for the cooperative observation of multiple moving targets (CMOMMT) task, a test-bed first introduced by Parker [8]. This scheme

[1] Robot Allocation through Coalitions using Heterogeneous Non-Cooperative Agents

mimics ant-behavior to regulate the distribution of sensors in proportion to that of the mobile targets. Dahl et al. [9] present a task allocation scheme based on "Vacancy Chains," a social structure modeled on the creation and filling up of vacancies in an organization. The Broadcast of Local Eligibility system (BLE) [10] system uses a Publish/Subscribe method to allocate tasks that are hierarchically distributed.

Market based task allocation systems have traditionally found favor with the software-agent research community. The inspiration for these systems stems from the Contract Net protocol [11]. Variations of the Contract Net protocol have found applications in numerous software-agent negotiation scenarios, these include [12], [13], [14] and [15].

Robotics researchers have designed a variety of market based control architectures for multi-robot tasks. Stentz and Dias [16] were the first to utilize a market-based scheme to coordinate multiple robots for cooperative task completion. This work introduced the methodology of applying market mechanisms to intra-team robot coordination as opposed to competitive inter-agent interactions in domains such as E-commerce. Laengle et al. [17] implemented the KAMARA system which uses a negotiation based task allocation scheme for controlling the different components of a complex robot. Caloud et al. [18] developed the GOPHER architecture that utilizes a centralized auction protocol to allocate tasks with a high level of commitment. Gerkey and Matarič [19] developed MURDOCH; a completely distributed auction-based task allocation scheme that utilized a publish/subscribe communication model. Tasks in MURDOCH are allocated via a single round, first price auction in a greedy fashion. M+, another auction based task allocation protocol was developed by Botelho and Alami [20]. The novelty of the M+ system lies in that it allows for dynamic task reallocation of subcomponents of complex tasks. Dias [21] designed the Traderbots architecture for multirobot control. Traderbots agents called traders are responsible for trading tasks via auctions. When an auction is announced agents compute bids based on their expected profit for the tasks, and the robots that can perform the tasks for the lowest price are awarded contracts.

As previously stated, a relatively unexplored problem in multi-robot systems is the allocation of multi-robot teams to different tasks (the ST-MR problem). Very recently Zlot and Stentz [22] have designed a scheme for complex task allocation for multi-robot teams. This scheme allows bidding on tasks at different levels of the decomposition hierarchy via the use of task trees. To the best of our knowledge, this is the only system apart from the one described here that deals with task allocation for multi-robot tasks.

Coalition formation is an established area of DAI research and many techniques for agent coalition formation have been proposed. Sandholm et al. [2] formally prove that finding the optimal coalition structure is an NP-Complete problem. They view the coalition structure generation process as a search of a coalition structure graph[2]. The problem is formulated as a search through a subset of the coalition structures and selection of the best coalition structure encountered. Fass [23] provides an Automata-theoretic view of agent coalitions

that discusses results that can be adapted to selecting groups of agents. Li and Soh [24] discuss the use of a reinforcement learning approach in which agents learn to form better coalitions over repeated iterations. Sorbella et al. [25] describe a mechanism for coalition formation based on a political society. A novel organisational approach was recently proposed by Abdallah and Lesser [26]. In this approach, an organization-based distributed polynomial time algorithm is proposed that makes use of reinforcement learning to optimize local decisions made by agents in the organization as the agents gain experience.

3 Agents vs. Robots

Despite the similarities between multi-agent and multi-robot systems, the algorithm translation from agents to robots is not straightforward. There are some inherent differences between software agents and robots and the environments in which they both operate. Software agents are simply code fragments whose capabilities correspond to software functionality and current data knowledge. Robots are tangible entities that occupy physical space and whose capabilities correspond to sensors, actuators, etc. Whereas software agents are programmed by individual programmers, robots are manufactured on a large scale. Software agents may dynamically alter their capabilities by downloading additional code, but robot capabilities remain static. Due to the flexibility that comes with software agents, DAI researchers make numerous assumptions while designing algorithms that do not hold when those algorithms are applied to robots. Besides these assumptions, robots must handle real world sensory noise, full or partial robot failures, and communication latency or loss of communications. All of these issues must be addressed before a multi-agent algorithm may be considered viable for robotic applications. The following section addresses these issues and suggests modifications to an existing multi-agent coalition formation algorithm.

4 The Multi-Robot Coalition Formation Algorithm

This section provides a brief overview of an existing multi-robot coalition formation algorithm that leverages a heuristic from the multi-agent coalition formation literature. Also, presented is the concept of coalition imbalance and its implications with respect to fault tolerance in multi-robot coalitions.

4.1 Shehory and Kraus' algorithm

Shehory and Krauss [27] designed a heuristic based algorithm for task allocation via agent coalition formation in Distributed Problem Solving (DPS) environments. The algorithm restricts the coalition structure space by restricting the size of individual coalitions to a fixed constant k. This heuristic fits well with the multi-robot task domain because very often task execution requires less than a fixed number of robots. The algorithm consists of two primary stages:

1. Calculate the coalition values thus enabling their comparison.
2. Agents determine via an iterative greedy process the preferred coalitions and form them.

Due to the differences between agents and robots outlined above, the current algorithm cannot be directly applied for multiple-robot coalition formation. The subsequent sections provide a cursory glance at some of these issues and the solutions [4]. The impact of extensive communication was shown to be severe enough to endorse relinquishing communincation in favour of additional computation when possible. The task format for multi-robot coalitions was modified to adequately represent additional constraints imposed by the multi-robot domain. The concept of coalition imbalance was introduced and its impact on the coalition's fault tolerance was demonstrated.

4.2 COMPUTATION vs. COMMUNICATION

Shehory and Kraus' algorithm [27] in its original form requires extensive communication and synchronization during the computation of coalition values. While this may be inexpensive for disembodied agents, it is often desirable to minimize communication in multiple-robot domains even at the expense of extra computation. The purpose of communication in Shehory and Kraus' algorithm is to allow the agents to continuously update each others capability values and to avoid redundant computations. Since robot capabilities typically remain static, the need for continuous updates is eliminated. Therefore, we altered the algorithm to allow each robot to evaluate all coalitions in which it was a member and eliminated the need for communication in the coalition evaluation stage.

4.3 TASK FORMAT

The multi-agent coalition formation algorithm [27] assumes that the agents have a capability vector, $< b_1^i, ..., b_r^i >$. Agents are allowed to exchange resources, therefore an agent coalition is deemed to have sufficient resources to perform a task if each element of the vector sum of the agent capabilities is greater than the vector of capability requirements for the task. Shehory and Kraus' algorithm assumes that the individual agents' resources are collectively available upon coalition formation. The formed coalition freely redistributes resources amongst the members. However, this is not possible in a multiple-robot domain. Robot capabilities correspond to sensors (camera, laser, sonar, or bumper) and actuators (wheels or gripper) which cannot be autonomously exchanged. This implies that simply possessing adequate resources does not necessarily create a multiple-robot coalition that can perform a task, other locational constraints have to be represented and met. These constraints were represented using a task constraint graph and the coalitions were validated using arc-consistency to check for constraint violations.

Using the Constraint Satisfaction Problem (CSP) formulation each candidate coalition is checked to verify if its coalition is feasible. After constraint checking

fewer coalitions remain for further evaluation. While additional overhead is incurred during constraint checking, this overhead is somewhat compensated for by the reduced number of coalitions.

4.4 COALITION IMBALANCE

Coalition imbalance or lopsidedness is defined as the degree of unevenness of resource contributions made by individual members to the coalition. This characteristic is not considered in other coalition formation algorithms. A coalition where one or more agents have a predominant share of the capabilities may have the same utility as a coalition with evenly distributed capabilities. Recall from Section 3 that robots are typically unable to redistribute their resources. Therefore coalitions with one or more dominating members (resource contributors) tend to be heavily dependent on those members for task execution. These dominating members then become indispensable. Such coalitions should be avoided in order to improve fault tolerance. Over reliance on dominating members can cause task execution to fail or considerably degrade. If a robot is not a dominating member (does not possess many sensors) then it is more likely that another robot with similar capabilities can replace this robot. The Balance Coefficient metric [4] quantifies the coalition imbalance level.

Simulation experiments were performed in the Player/Stage [28] simulation environment to demonstrate the effect that the balance coefficient would have on fault tolerance of robot teams. The algorithm was then transferred to actual robots in which the box-pushing task was used as a proof of concept experiment.

The translation of the simulation experiments to real world robot experiments required a few adjustments to account for problems of slippage and mapping of capabilities to real numbers. However, the experiments successfully establish that the algorithm performs satisfactorily with real robots.

5 Coalition Formation as a Matching Problem

The title "coalition formation" is to some degree a misnomer in that it implies that the problem is simply the partitioning of the set of robots into teams or coalitions. In reality, coalition formation as defined in game theory and DAI is only half of the problem, the other half of the problem is the assignment of coalitions to individual tasks so as to maximize the overall system utility. At first glance these two phases seem to be independent but closer examination reveals a strong link. The coalition value is determined largely by the utility of the task assigned to that coalition. Therefore, the coalition formation process has to consider all tasks while considering possible coalition structures (containing coalitions to a size k). This is what makes coalition formation a harder problem than a simple instance of the Set Partitioning problem where the values of subsets are fixed.

Our current coalition formation algorithm [4] greedily chooses the best coalition-task pair. Although this greedy approach yields a fast solution, it also detracts

from the solution optimality. Therefore, we propose to formulate the problem as a matching problem and simultaneously consider all the tasks and coalitions.

5.1 THE MATCHING PROBLEM

A matching in a graph $G = (V, E)$ is a subset M of the edges E such that no two edges in M share a common end node. If the graph's edges have an associated weight, then a maximum weighted matching is a matching such that the sum of the matching edge weights is maximized.

A graph is *bipartite* if it has two kinds of nodes and edges exist only between two nodes of a different kind. Matchings in bipartite graphs can be computed more efficiently than matchings in general (non-bipartite) graphs.

A maximum cardinality matching is a matching with a maximum number of edges. If the graph edges have an associated weight, then a maximum weighted matching is a matching such that the sum of the matching edge weights is maximized. A maximum weighted maximum cardinality matching is a maximum cardinality matching with the maximum weight.

5.2 MATCHING COALITIONS AND TASKS

Once the agents have created a list of coalitions and evaluated all coalitions as discussed in Section 4, the coalitions and the tasks can be represented as the nodes of a bipartite graph, as shown in Fig 1. Once the tasks and coalitions are represented as nodes of a bipartite graph, the selection of coalitions can be formulated as a matching problem.
Three types of problems can arise depending upon what parameter is to be maximized:

1. Maximize the number of tasks assigned: This is an instance of the *maximal cardinality bipartite matching problem.* The best known algorithm for solving this problem was developed by Hopcroft and Karp [29].
2. Maximize the overall utility: This is an instance of the *maximum weighted matching problem.* The best known algorithm for solving this problem was provided by Tarjan [30].
3. Maximize number of tasks and for this maximal number, maximize the utility: An instance of the *maximum weighted maximum cardinality matching problem.* The best known algorithm for solving this problem was provided by Kuhn [31].

The problem with formulating the coalition formation as a matching problem is that the chosen coalitions may overlap, i.e. the members of one coalition in the maximal matching may be present in another. Since at the moment only disjoint coalitions are being considered, we need to ensure that the final chosen coalitions do not overlap. This is accomplished by iteratively computing the maximal matching using the appropriate matching algorithm as listed above. The edges of the matching are arranged in ascending order (in case of weighted

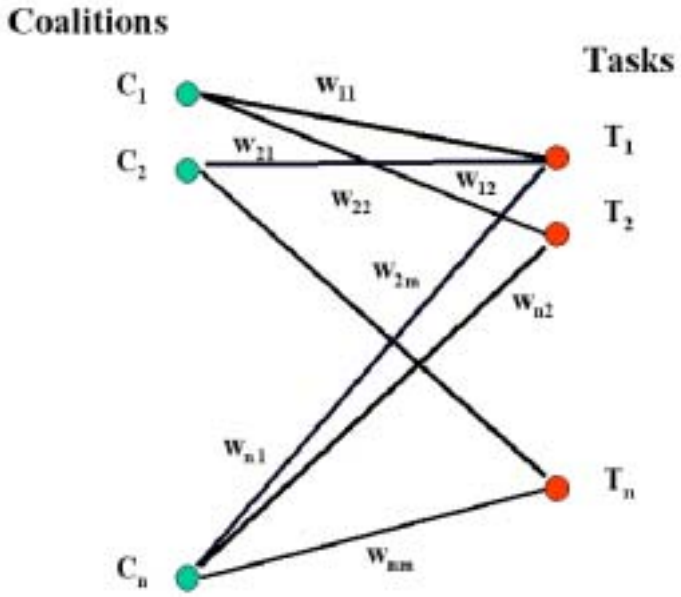

Fig. 1. Matching coalitions with tasks.

maximal matching). Each edge (coalition-task pair) is selected in order, and all coalitions with members common with the selected coalition are deleted from the graph. Additionally, all the edges incident on those coalitions are deleted from the list of edges in the current maximal matching. This edge deletion continues until all the edges are exhausted and the list of edges is empty. A new maximal matching is then computed with the remainder of the coalitions and tasks. The maximal matching for the resulting graph is then recomputed and the entire process is repeated until either all the coalitions are deleted or all the tasks have been assigned.

6 Task Environments

The suitability of solutions to the coalition formation problem depends to a large degree on the nature of the task environments. Solutions that are appropriate for a particular task environment may not be appropriate for a different task environment. This section outlines some approaches for coalition formation in different task environments. It should be noted that no assumptions about the nature of the tasks are being made, only about the manner in which tasks are introduced into the system.

6.1 STATIC TASKS

If the tasks are independent and task utilities do not change with time, then the task environment is called *static*. Robots in static environments must allocate

themselves amongst the various tasks only once, i.e. once a given set of tasks are allocated, the algorithm terminates. This is the simplest task environment and the algorithm described in Section 4 can be used to form coalitions.

6.2 PRECEDENCE ORDERED TASKS

There may be occasions where a specific task t_j cannot be performed unless another specific task t_i has already been satisfied. This is generalized by a partial precedence order between the tasks, $t_1 \preceq t_2 \ldots \preceq t_n$, where $t_i \preceq t_j$ means that t_i is the predecessor of t_j in the performance order. Consider the blocks world problem as shown in Figure 2, the task is to arrange a set of blocks starting from an initial configuration into a final configuration. Assuming that the blocks are too heavy for an individual robot to transport, each block must be moved by a coalition of robots. In Figure 2 blocks D_1, D_2 and D_3 may be transported at different times (as long as D_1 and D_2 are placed before D_3). Therefore the final configuration may be attained by formation of three overlapping coalitions, each responsible for the placement of one of the blocks D_1, D_2, D_3. These three coalitions may have robots in common because the tasks performed by these coalitions may be performed at different times.

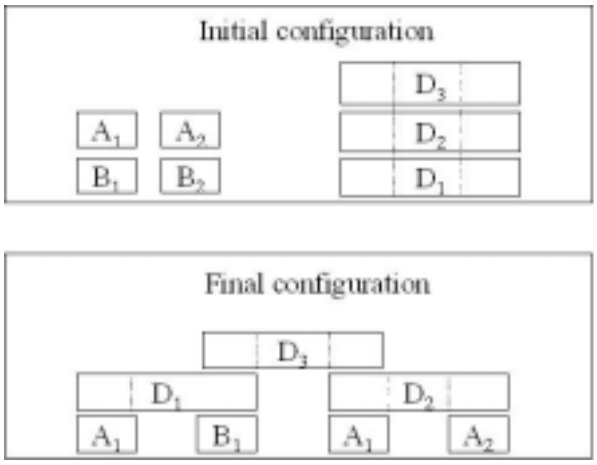

Fig. 2. The Blocks world [26].

Since these tasks have to be executed in order, the coalition members may overlap. However, closer examination reveals that the twin tasks of arranging blocks D_1 and D_2 are independent of each other, i.e. they may be executed in parallel or in any random order. Therefore, it would be more efficient to assign disjoint coalitions to these tasks to allow for the parallel arrangement of D_1 and D_2. This idea can be generalized to a sequence of precedence ordered tasks where some intermittent sub-sequences may consist of independent tasks. The most efficient solution in this case would be to execute the tasks in such a

subsequence in parallel by assigning disjoint coalitions to as many tasks in the subsequence as possible. Subsequences of tasks for which interdependencies do exist may be performed using overlapping coalitions.

The Algorithm: The idea is to find the largest subset of tasks consistent with the task ordering that can be executed with the currently available set of robots. Instead of evaluating individual coalitions against all the tasks, sets of coalitions must now be examined. Unlike the algorithm in Section 4 where one coalition was assigned to a task in each iteration. In the precedence order case, the agents may form several coalitions in each iteration. Formally the algorithm in this case is:

1. Find all tasks that have no unfulfilled pre-requisite tasks. Call this list of candidate tasks the candidate list C_l.
2. Evaluate all coalition structures comprising of coalitions of size k or less with respect to the tasks in C_l.
3. Choose the coalition structure with the highest utility of coalitions and assign each coalition to its designated task.
4. Remove the tasks executed from C_l and add any new tasks that are now eligible for execution.
5. Check if any robots have completed their task and if so add them to the list of available robots.
6. Repeat until no more tasks remain to be executed.

6.3 DYNAMIC TASKS: The RACHNA Architecture

A common feature of the market based systems discussed in Section 2 is that all these systems require the robots to bid on the tasks. While this approach is adequate for allocating single-robot (SR) tasks, it does not allow for the gathering of the global information necessary to allocate resources to multi-robot (MR) tasks. The bidding process is central to determining the auction outcome. Therefore when dealing with complex tasks, the bidder should have a global view of the available resources. Previous systems require a robot to bid without incorporating any knowledge of the status of other robots (i.e. busy or idle). We propose a system, namely RACHNA[2], in which the bidding is reversed. The auction is performed by the tasks for the individual robot services. This allows for the bidding to be performed with the global information necessary for coalition formation. There are two types of software agents that are involved in the task allocation:

1. **Service Agents:** The Service Agents are the mediator agents through which the tasks must bid for a service. RACHNA requires that each robot has a set of services or roles that it is capable of performing. The roles are determined by the individual sensor and behavioral capabilities resident on each robot.

[2] Robot Allocation through Coalitions using Heterogeneous Non-Cooperative Agents

There is one service agent for each service type that a robot can provide. A service agent may communicate with any of the robots that provide the particular service to which the agent corresponds. For example, the foraging service agent may communicate with all robots that currently have sensor capabilities i.e. (camera, gripper) to perform the foraging service. Service agents reside on any one of the robots that are capable of providing the service. Thus, the global information concerning the task is acquired in a decentralized manner through the use of service agents.

2. **Task agents:** These agents place offers on behalf of the tasks so as to acquire the necessary services. The task agents communicate only with the service agents during negotiations. Once the task has been allocated, the task agent may communicate directly with the robots that have been allocated for the task.

Figure 3 provides an overview of an example RACHNA architecture implementation.

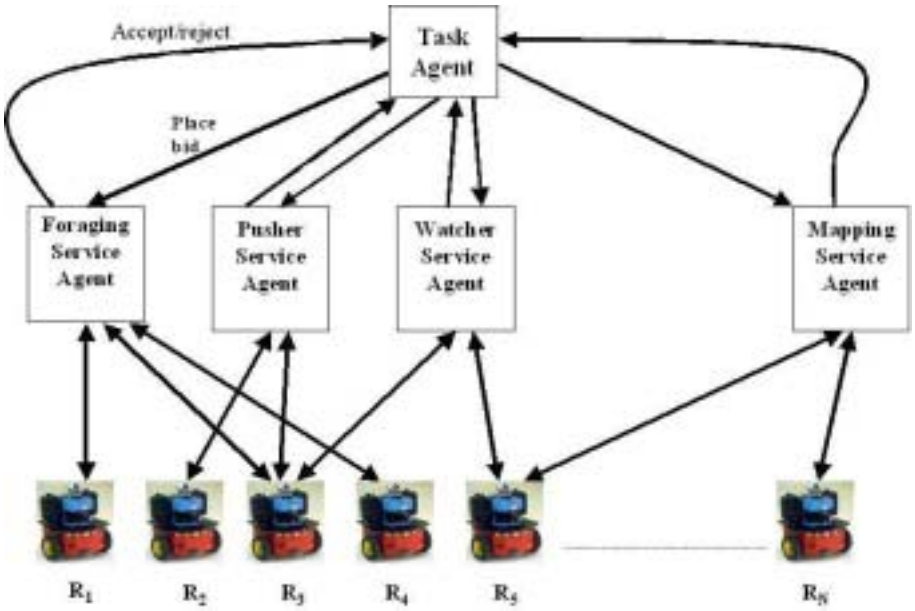

Fig. 3. An example RACHNA Implementation.

We propose an economy where the tasks are represented by task-agents that are bidding for the services of the individual robots. The economy has a set of robots $R_1, R_2..., R_N$ where each robot is equipped with sensor capabilities that enable it to perform various services such as pushing, watching, foraging, etc. The tasks are assumed to be decomposable into the sub-task behaviors (roles) that they require. For example, in the box-pushing task as defined in [32], two pusher

sub-task roles are required and one watcher sub-task role is required. Each role is represented by a service agent that is responsible for negotiating with the robots with the desired capability. The roles may be implemented through the use of behavior sets as defined in [5]. The negotiation proceeds as follows, on arrival of a task with a certain utility, a new task agent is created that considers the roles required for task execution and communicates an offer to the respective service agents. The initial offer for each role is an equal distribution of the task utility amongst the robots engaged in the task. The contacted service agents then attempt to find a suitable robot that is currently receiving a lower utility than has been offered. If there is no such robot with the required capabilities, the service agent sends a reject message to the task agent along with the minimum utility that would have to be offered for the provided service. If all the service agents reject the task agent's offer, then the desired resources cannot be allocated and the task agent must wait until the task utility increases (utility may be modeled as an increasing function of time) or until the task agent receives a signal from the service agent indicating that the service has become cheaper. If all service agents accept the offer, then the task is successfully allocated. The interesting case occurs when some service agents accept and others reject the offer. The twin quantities of Slack Utility and Utility Requirement are defined in order to handle this situation.

Slack Utility: The slack utility of a task is defined as the excess utility offered to the service agents for the individual services. In other words, it is the maximum utility that can be withdrawn from an offer which will still enable the task to retain the current set of services. Formally, if the current set of services that have accepted require a minimum of U_{min} to continue to stay committed to the task and if the previous combined bid for these services was U_p then the Slack Utility (SU) is given by:

$$SU = U_p - U_{min} \tag{1}$$

Utility Requirement: The utility requirement of a task is defined as the minimum additional utility needed to acquire remaining services required for the task. Formally, the current set of services that have refused to commit to the task require a minimum of U_{min} to do so, and the previous combined bid for these services is U_p, then the Utility Requirement (UR) is given by:

$$UR = U_{min} - U_p \tag{2}$$

After the receipt of the accept/reject messages from the service agents, the task agent has sufficient global information to determine whether the task resources can be acquired. When the task agent receives acceptances from some agents and has enough slack utility to purchase more services, it calculates the slack utility available to the task and evaluates whether this slack is sufficient (i.e. if $SU > UR$). If enough SU exists, the entire extra slack is offered to the individual service agents one at a time. Note that although the Slack Utility

may seem sufficient, the services may still be refused if the other tasks increase their offers to the individual robots in response to the threat of losing a robot. If the current Slack Utility proves insufficient then the services cannot be acquired and the task agents wait until the task utility increases to the desired level or until the service agents indicate that the service is now available at a lower price. These tasks are placed in the service agent's task queue to wait until some robots are free or have completed their designated task (and have become idle). Robots will inform their respective service agents as soon as they are free, the waiting tasks in the service queue will then be sent a signal to restart the auction process. Care must be taken to disallow the different service agents from competing with each other for different services on the same robot for the same task.

Once all the services have been acquired, the robots are sent a 'green' signal to leave their current task and start performing the new task. A robot only stops performing its current task if it has received a 'green' signal from the new task. The negotiation process is carried out asynchronously via a multi-threaded communication mechanism.

The interaction between the robots and service agents is fairly involved. If a robot receives an offer that is better than its current utility (plus the penalty it would have to pay for decommiting from its current task), a robot will send a bargaining message to its current task agent asking for more utility. The task agent would then reply with one of two possible messages:

- An offer that matches the robot's current best offer.
- A regret message indicating that it cannot offer a utility that matches the robot's current best offer.

If the task agent sends the first message then the robot's utility is increased and the robot is 'happy' with its new salary and continues working on the task. If the second message is sent, the robot decommits and leaves for the task that offered a better pay package.

Note that this work does not focus on the decomposition of tasks. Instead we assume a task is pre-decomposed into the required set of behaviors or roles. If a given behavior decomposition cannot be allocated (i.e. the service agents return a negative response), alternative decompositions may be considered.

UTILITIES AND PENALTIES: One difficulty with the economy based approach that RACHNA employs is that the resulting teams are highly dependent on the initial utilities assigned to various tasks. However, this may not be an entirely undesirable property, while it makes the system sensitive to initial utilities, it also empowers the user to prioritize tasks by varying the task utilities. The payoff distribution to individual team members is handled through the negotiation process.

Different multi-robot tasks require different levels of commitment. For example, in a multi-robot foraging task, if one of the foragers decides to decommit, the task can still be performed, albeit more slowly. However, in tightly coupled

tasks such as construction or box-pushing, decommiting can have a severe impact on task performance. RACHNA provides the user with the ability to define the appropriate level of commitment via the use of penalties. For example, in the case of tightly coupled tasks, the penalty of leaving a task should be high in order to discourage agents from leaving after accepting and rendering the task incomplete.

If a robot loses control of a sensor or actuator in RACHNA, the system allows for graceful performance degradation. If a sensor failure occurs a robot may still be capable of performing an alternative behavior, due to the existence of the mapping from sensor capabilities to behavioral capabilities. Consider a robot that is capable of performing the foraging and watcher behaviors. If the robot's gripper is damaged, it will be unable to execute the foraging behavior but it may still be able to perform the watcher behavior. The foraging service agent system simply deletes the robot from the list of foragers and in future auctions this robot will not receive offers relating to the foraging behavior. The robot will not be deleted from the list of watchers because the relevant sensors for watching (camera) are still intact. Thus, the system allows for graceful degradation in performance. It should be noted that the system makes no attempt to detect faults.

COMMUNICATION: Communication as highlighted in Section 4.2 as a very important issue in Multi-Robot System (MRS) system design. Communication failures are fairly common in robotic systems, hence it is important that the system should be robust to communication failures. RACHNA requires only that the robots within a defined communication range respond with accept/reject messages. So while the system has fewer robots to choose from when a robot undergoes a communication failure, the robot's commmunication failure does not debilitate the system. If a robot fails during task execution, the task agent simply attempts to bid for another robot to perform the service.

DISCUSSION: This section introduced RACHNA, a novel negotiation-based approach to the multi-robot coalition formation problem that takes advantage of this inherent redundancy in robot capabilities by grouping the robots according to their behavioral capabilities. The reversal of the bidding process with tasks bidding on robots allows for acquisition of the global resource information necessary for dynamically solving the coalition formation problem.

As outlined in Section 3, there are some inherent differences between the multi-agent and multi-robot domains. One of the most prominent of these differences is the level of redundancy in multi-robot and software-agent capabilities. Whereas software-agents are simply code fragments programmed by individuals, robots are manufactured on a large scale. Therefore, robots are more likely to have greater redundancy in their sensor/actuator capabilities. Indeed, almost any modern day robotics facility would have a number of robots with identical capabilities. To the best of our knowledge, RACHNA is the first system to

leverage this redundancy to enable a more tractable formulation of the coalition formation problem.

7 Conclusion and Future Work

This paper addresses the problem of a heterogeneous set of robots coalescing into teams to perform a set of complex (multi-robot) tasks. The paper focuses on illustrating how the coalition formation problem is formulated as an instance of the matching problem which can yield solutions closer to the optimal. Finally, the paper discusses coalition formation techniques for task environments with static, precedence ordered, and dynamic tasks. RACHNA, a novel market based system is presented that enables dynamic reconfiguration of the the coalitional configuration on introduction of new tasks into the task environment. In conclusion, this work provides the outline for a generic, task independent framework for the formation of multi-robot coalitions in different task environments.

This work is a precursor to a more thorough investigation into the multi-robot coalition formation problem. Even though experiments have been conducted to demonstrate that the algorithm described in Section 3 can be used with real robots, the experiments do not prove that the algorithm scales because the experiments were performed with a total of three robots. Therefore further experiments will be conducted with a larger number of robots and a heterogeneous set of tasks. The prominent contribution of this paper is the RACHNA system which is currently being implemented. Experiments will be performed with the RACHNA system to demonstrate its application to a dynamic task environments and to test its robustness to variations in incoming task rate and task utilities. Also experiments will be conducted to demonstrate the fault tolerance of the system to both partial and complete robot failure. Another aspect of the system that requires study is the extent of communication involved during the negotiation process. This would reveal important results with regard to the system scalability. These experiments will first be conducted via simulations and then be conducted on real robots.

References

1. Gerkey, B., Matarić, M.J.: A framework for studying multi-robot task allocation. In: Proceedings of the Multi-Robot Systems: From Swarms to Intelligent Automata. Volume 2. (2003) 15–26
2. Sandholm, T.W., Larson, K., Andersson, M., Shehory, O., Tomhe, F.: Coalition structure generation with worst case guarantees. Artificial Intelligence **111** (1999) 209–238
3. Gerkey, B., Matarić, M.J.: Are (explicit) multi-robot coordination and multi-agent coordination really so different? In: Proceedings of the AAAI Spring Symposium on Bridging the Multi-Agent and Multi-Robotic Research Gap. (2004) 1–3
4. Vig, L., Adams, J.A.: Issues in multi-robot coalition formation. In: Proceedings of Multi-Robot Systems. From Swarms to Intelligent Automata. Volume 3. (2005) 15–26

5. Parker, L.E.: Alliance: An architecture for fault tolerant multi-robot cooperation. IEEE Transactions on Robotics and Automation **14** (1998) 220–240
6. Brooks, R.A.: A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation **2** (1986) 14–23
7. Low, K.H., Leow, W.K., M. H. Ang, J.: Task allocation via self-organizing swarm coalitions in distributed mobile sensor network. In: Proceedings of the American Association of Artificial Intelligence. (2004) 28–33
8. Parker, L.E.: Cooperative robotics for multi-target observation. Intelligent Automation and Soft Computing, special issue on Robotics Research at Oak Ridge National Laboratory **5(1)** (1999) 5–19
9. Dahl, T.S., Matarić, M.J., Sukhatme, G.S.: Multi-robot task-allocation through vacancy chains. In: Proceedings of IEEE International Conference on Robotics and Automation. (2003) 14–19
10. Werger, B., Matarić, M.J.: Broadcast of local eligibility: Behavior based control for strongly cooperative multi-robot teams. In: Proceedings of Autonomous Agents. (2000) 21–22
11. Smith, R.G.: The contract net protocol: High level communication and control in a distributed problem solver. IEEE Transactions on Computers **C-29** (1980) 1104–1113
12. Sandholm, T.: An implementation of the contract net protocol based on marginal cost calculations. In: Proceedings of the Eleventh National Conference on Artificial Intelligence. (1993) 256–262
13. Collins, J., Jamison, S., Mobasher, B., Gini, M.: A market architecture for multi-agent contracting. Technical Report 97-15, University of Minnesota, Dept. of Computer Science (1997)
14. Sandholm, T., Lesser, V.: Advantages of a leveled commitment contracting protocol. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence. (1996) 126–133
15. Sycara, K., Zeng, D.: Coordination of multiple intelligent software agents. International Journal of Intelligent and Cooperative Information Systems **5** (1996) 181–211
16. Stentz, A., Dias, M.B.: A free market architecture for coordinating multiple robots. Technical Report CMU-RI-TR-01-26, The Robotics Institute, Carnegie Mellon University (1999)
17. Laengle, T., Lueth, T.C., Rembold, U., Woern, H.: A distributed control architecture for autonomous mobile robots. Advanced Robotics **12** (1998) 411–431
18. Caloud, P., Choi, W., Latombe, J.C., Pape, C.L., Yim, M.: Indoor automation with many mobile robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (1990) 67–72
19. Gerkey, B.P., Matarić, M.J.: Murdoch: Publish/subscribe task allocation for heterogeneous agents. In: Proceedings of Autonomous Agents. (2000) 203 – 204
20. Botelho, S.C., Alami, R.: M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In: Proceedings of IEEE International Conference on Robotics and Automation. (1999) 1234 – 1238
21. Dias, M.B.: TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments. PhD thesis, Robotics Institute, Carnegie Mellon University (2004)
22. Zlot, R., Stentz, A.: Complex task allocation for multiple robots. In: Proceedings of the IEEE Conference on Robotics and Automation (ICRA). (2005) 67–72

23. Fass, L.F.: Automatic-theoretic view of agent coalitions. Technical Report WS-04-06, American Association Of Artificial Intelligence Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems (2004)
24. Li, X., Soh, L.K.: Investigating reinforcement learning in multiagent coalition formation. Technical Report WS-04-06, American Association Of Artificial Intelligence Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems (2004)
25. Sorbella, R., Chella, A., Arkin, R.: Metaphor of politics: A mechanism of coalition formation. Technical Report WS-04-06, American Association Of Artificial Intelligence Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems (2004)
26. Abdallah, S., Lesser, V.: Organization-Based Cooperative Coalition Formation. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Techonology, IAT. (2004) 162–168
27. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence Journal **101** (1998) 165–200
28. Gerkey, B.P., Vaughan, R.T., Stoy, K., Howard, A., Sukhatme, G.S., Matarić, M.J.: Most valuable player: A robot device server for distributed control. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2001) 1226–1231
29. Hopcroft, J.E., Karp, R.M.: An n 5=2 algorithm for maximum matching in bipartite graphs. SIAM Journal of Computing **2** (1973) 225–231
30. Tarjan, R.E.: Data Structures and Network Algorithms. SIAM publications (1983)
31. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly (1955) 83–97
32. Gerkey, B.P., Matarić, M.J.: Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2002) 464 – 469