Multi-Robot Coalition Formation

Lovekesh Vig and Julie A. Adams, *Senior Member, IEEE*

***Abstract*—As the community strives towards autonomous multi-robot systems, there is a need for these systems to autonomously form coalitions to complete assigned missions. Numerous coalition formation algorithms have been proposed in the software agent literature. Algorithms exist that form agent coalitions in both super additive and non-super additive environments. The algorithmic techniques vary from negotiation-based protocols in multi-agent system (MAS) environments to those based on computation in distributed problem solving (DPS) environments. Coalition formation behaviors have also been discussed in relation to game theory. Despite the plethora of MAS coalition formation literature, to the best of our knowledge none of the proposed algorithms have been demonstrated with an actual multi-robot system. There exists a discrepancy between the multi-agent algorithms and their applicability to the multi-robot domain. This paper aims to bridge that discrepancy by unearthing the issues that arise while attempting to tailor these algorithms to the multi-robot domain. A well-known multi-agent coalition formation algorithm has been studied in order to identify the necessary modifications to facilitate its application to the multi-robot domain. This paper reports multi-robot coalition formation results based upon simulation and actual robot experiments. A multi-agent coalition formation algorithm has been demonstrated on an actual robot system.**

***Index Terms*—Coalition formation, coalition imbalance, task allocation.**

I. Introduction

SOFTWARE agent cooperation is a well-studied area in Distributed Artificial Intelligence (DAI). An agent may be motivated to cooperate if it encounters tasks that are difficult to accomplish and it is often beneficial to assign a group of agents to a task. This becomes necessary when a single agent cannot perform the tasks. A related research area is multi-agent coalition formation. Coalition formation partitions the set of agents into different teams where each team is assigned a task and is called a coalition. Choosing the optimal coalition from the set of all possible coalitions is generally an intractable problem due to the size of the coalition structure space [1]. Algorithms exist that yield solutions within a bound from the optimal and are tractable [2]–[4]. Gerkey and Matarić [5] indicate that despite the existence of various multi-agent coalition formation algorithms, none of these algorithms have been demonstrated in the multi-robot domain. The reason for this discrepancy is that these algorithms make underlying assumptions that are not applicable to the multi-robot domain. This paper identifies these assumptions and provides modifications to a multi-agent coalition formation algorithm to enable application to the multi-robot domain.

Manuscript received February 9, 2005; revised November 11, 2005. This work was supported by Vanderbilt University, Nashville, TN..

The authors are with the Electrical Engineering and Computer Science Department, Vanderbilt University, Nashville, TN 37235 USA (e-mail: lovekesh.vig@vanderbilt.edu).

Digital Object Identifier 10.1109/TRO.2006.878948

Researchers have been grappling with the problem of multi-robot task allocation for some time. Typically, the task allocation problem divides a task into indivisible subtasks and assigns robots to each subtask. This is referred to as the single robot-single task problem (SR-ST) [5]. Various approaches to this problem have been proposed. Parker's ALLIANCE [6] architecture uses motivational behaviors to monitor task progress and dynamically reallocate tasks. The MURDOCH [7] and BLE [8] systems use a Publish/Subscribe method to allocate tasks that are hierarchically distributed. Dias' [9] Traderbots architecture for multi-robot control utilizes trader agents that trade tasks via auctions.

This paper addresses the more difficult multiple robot-multiple task problem (MR-MT) [10] where the objective is to assign a robot team to a task so that the system's overall utility is maximized. This problem, also called the coalition formation problem, is more complex than the SR-ST problem on which current task allocation schemes focus. Gerkey and Matarić [5] formulate the task allocation problem as an instance of the Optimal Assignment Problem (OAP). While outlining some deficiencies in this formulation, they draw attention to the fact that the OAP framework does not address the coalition formation problem. They also draw attention to the inherent intractibility of searching for the optimal coalition, and the difficulties with multi-robot task allocation such as the requirement that the coalition structures be dynamic in order to deal with changing task requirements.

Coalition formation for multi-robot tasks remains an open problem and is the focus of this work. The solution lies in the formation of multi-robot teams or coalitions appropriate for a set of tasks. Each robot coalition is assigned a particular task and is responsible for its execution. The optimal solution to the coalition problem is NP-hard [1], however, problems very close to the coalition formation problem have been extensively studied, (i.e., Set Partitioning and Set Covering) and many heuristics for approximate solutions have been devised [11]–[14]. Game theorists and economists have studied the coalition formation problem with regard to market based selfish agents. They have investigated various types of equilibrium that lead to stable coalitions for selfish agents. In fact, coalition theory is now considered a field in its own right.

DAI researchers have built upon game theory and theoretical computer science to produce practical solutions to the multi-agent coalition formation problem. DAI has made considerable progress in the area of multi-agent coalition formation. Despite this progress and the numerous software agent coalition formation algorithms, to the best of our knowledge none of these algorithms have been demonstrated in the multi-robot domain. The software agent algorithms do not directly transfer to multi-robot systems, because robotic systems must address real world issues such as physical limitations of sensor and actuator capabilities, robot malfunctions, dynamic environments, and communication failure. Thus there exists a discrepancy between the multi-agent coalition formation literature and its application to the multi-robot domain. This work aims to bridge this discrepancy.

Current multi-robot task allocation schemes assume that the entire set of robots is available for task execution and that communication is always possible between the robots (or that the system can provide motivational feedback). These assumptions need not always hold, a set of tasks may be located sufficiently far from one another so that the best solution is to dispatch a robot team to each designated task area to complete the task autonomously.

It may be that once at the task site the team members are only able to communicate amongst themselves and cannot communicate with the rest of the robots. The robots then coalesce into teams responsible for each task. The focus of this work is to investigate the various issues that arise while forming multi-robot coalitions utilizing existing multi-agent coalition formation algorithm. Specifically, Shehory and Kraus' multi-agent task allocation algorithm [3] is modified to operate in the multi-robot domain. This algorithm was chosen because it is designed for DPS environments, has an excellent real-time response, and has been empirically shown to provide results within a bound from the optimal [3].

This paper is organized as follows. Section II provides related work. Section III presents an overview of Shehory and Kraus' algorithm. Section IV identifies issues that entail modification of the current coalition formation algorithm. Experimental results are provided in Section V. Finally, Section VI discusses the conclusions.

II. Related Work

DAI research is divided into two basic classes: Cooperative Distributed Problem Solving (CDPS) and Multi-Agent Systems (MAS). CDPS research considers how a particular problem can be distributed across a number of modules or nodes. MAS research is concerned with coordinating intelligent behavior among a collection of autonomous, heterogeneous, intelligent agents. Global control and globally consistent knowledge may not exist in MAS. MAS agents are assumed to be self-interested; trying to achieve their own goals and maximizing their own personal payoff. CDPS agents have a common goal and attempt to maximize the system's global payoff. MAS designers control their own agents. These classes highlight the extreme poles in the DAI research spectrum. DAI researchers have worked to construct agents that are able to cooperate during task execution, thus increasing either their individual or system benefits.

The coaliton formation problem has been studied in both the MAS and CDPS environments. Environments with heterogeneous agents require protocols that must be agreed upon and enforced by the designers. The need for protocol increases further with coalition formation. Shapely and Shubik [15] noted "in situations where not every cooperative demand by every coalition can be satisfied, some constraints must be placed on coalitional activity lest it be trapped in an endless loop of rejected suggestions for coalition formation." This conclusion lead to the concept of stability and notions of equilibrium. Kraus *et al.* [16] provide agents with negotiation mechanisms that enable cooperation. Their work deals with time constraints and the resulting agreement efficiency.

Smith [17] presented the Contract Net Protocol (CNP) where agents attempt to satisfy a task by dividing it into sub-tasks and sub-contract each sub-task to another agent via a bidding mechanism. The CNP allocates tasks to single agents and a task partitioning procedure is necessary. Numerous multi-robot task allocation schemes have been inspired by the CNP [17]. Sandholm and Lesser [18] developed a coalition formation model for bounded rational agents and presented a general classification of coalition games. Their model assumes that the coalition value[1] depends on the computation time; however, all possible coalitions are considered when computing a stable solution. Shehory and Kraus [4] proposed coalition algorithms that distribute joint payoffs to members in Non-Super-additive environments. The same paper presents an alternative protocol offering a weaker form of stability with polynomial running time. However, in both cases, no bound from the optimal is guaranteed.

As previously mentioned, a relatively unexplored problem in multi-robot systems is the allocation of multi-robot (complex) tasks. Recently, Zlot and Stentz [19] designed a complex task allocation scheme for multi-robot teams that allows task bidding at different levels of the decomposition hierarchy via task trees. The ASyMTRe system [20] is the first system that autonomously generates decompositions of complex multi-robot tasks. ASyMTRe is based on the theory of information invariants [21] and examines the required connections between the perceptual, environmental, and communication schemas to generate different task decompositions. The appropriate task decomposition was utilized to enable multi-robot team formation [22]. However, the system does not address the decision problem of which team to assign to a particular task. To the best of our knowledge, these are the only other systems that deal with the allocation of multi-robot tasks.

Sandholm *et al.* [1] prove that finding the optimal coalition structure is an NP-Complete problem. They view the coalition structure generation process as a search of the coalition structure graph. The problem requires a search through a subset of the coalition structure space and the selection of the best coalition structure encountered.

Sandholm *et al.* [1] further prove that in order to establish a bound on the quality of the generated coalition structure, it is necessary and sufficient that the search algorithm must search the bottom two levels of the coalition structure graph. The suggested algorithm involves an initial search through these two levels and then a time bounded breadth first search from the top of the coalition structure graph. This was previously the only anytime algorithm that established a worst case bound, however, Dang and Jennings [23] have designed an algorithm that improves on Sandholm *et al.*'s algorithm, analyzing fewer coalitions while establishing small bounds from the optimal.

Shehory and Kraus [3] proposed a task allocation algorithm via coalition formation for CDPS environments. This algorithm limits the coalition sizes and uses a greedy heuristic to yield a coalition structure that is provably within a bound the of best

[1]The coalition value for coaltion C is the utility obtained when C performs the task that yields the highest utility for C.

solution given the limit on the number of agents (see [3] for a detailed complexity analysis). This algorithm is directly relevant to the work described in this paper and modifications to this algorithm are described in Section IV.

Currently a plethora of available agent coalition formation approaches have been proposed in the literature [24]–[27]. However, none of the algorithms have been demonstrated in the multi-robot domain. Some reasons for this include the added communication costs, the unreliability of both communication and robots in real world domains, and the non-transferability of resources between robots. This paper investigates these issues and offers solutions for tailoring a multi-agent coalition formation algorithm to the multi-robot domain.

III. Shehory and Kraus' Algorithm

Shehory and Kraus' [3] algorithm is designed for task allocation via software agent coalition formation in DPS environments. This heuristic based algorithm fits well with the multi-robot domain. This Section provides an algorithmic description along with necessary modifications required for application to the multi-robot domain.

A. Assumptions

Assume a set of n agents, $A = A_1, A_2, \ldots, A_n$. The agents communicate with each other and are aware of all tasks to be performed. Each agent A_i has an p-dimensional vector of real non-negative capabilities, $B_{A_i} = \langle b_1^{A_i}, b_2^{A_i}, \ldots, b_p^{A_i} \rangle$, where each capability is a property that quantifies the ability to perform an action. An evaluation function is attached to each capability type that transforms capability units into monetary units. It is assumed that there is a set of m independent tasks, $TS = t_1, t_2, \ldots, t_m$. Each task t_l has an p-dimensional capability requirement vector $B_{t_l} = \langle b_1^{t_l}, \ldots, b_p^{t_l} \rangle$. The utility gained from performing the task (or taskvalue) depends on the capabilities required for execution. A coalition is a group of agents that decide to cooperate to perform a common task and each coalition performs a single task. A coalition C has a p-dimensional capability vector B_c representing the sum of the capabilities that the coalition members contribute to this specific coalition. A coalition C can perform a task t_l only if the t_l's capability requirement vector B_{t_l} satisfies $\forall 0 \leq u \leq p$, $b_u^{t_l} < b_u^C$.

B. The Algorithm

The algorithm consists of two primary stages:

1) Calculate the coalitional values for comparison.
2) Determine, via an iterative greedy process, the preferred coalitions and form them.

Stage one is directly relevant to this work. During this stage the evaluation of coalitions is distributed among the agents via extensive message passing, which requires considerable inter-agent communication. After this stage, each agent has a list of coalitions for which it calculated coalition values. Agents also have all necessary information regarding the coalition memberships' capabilities. Each agent then calculates the coalition values by:

1) determining the necessary capabilities for each task, $t_i \in TS$ by comparing the required capabilities to the coalition capabilities;
2) calculating the best-expected task outcome of each coalition and choosing the coalition yielding the best outcome.

IV. Issues in Multi-Robot Coalition Formation

The Section III algorithm yields results that are close to optimal, however, the presented algorithm cannot be directly applied to multi-robot coalition formation. This section identifies issues that must be addressed when the algorithm is applied to the multi-robot domain.

A. Computation Versus Communication

Shehory and Kraus' algorithm [3] 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 multi-robot domains, even at the expense of extra computation. In this work, each agent assumes responsibility for evaluating all coalitions in which it is a member, thereby eliminating the need for communication. It is necessary to analyze how each robot's computational load is affected. An added assumption is that a robot has *a priori* knowledge of all robots and their capabilities [3]. Robot capabilities do not typically change, therefore, this is not a problem unless a partial or total robot failure is encountered, i.e., [28]. The total space of coalitions examined includes all coalitions of sizes less than or equal to the maximum allowed coalition size (k). Suppose there are n identical robots with a perfect computational load distribution, then the number of coalitions each robot must evaluate with communication is

$$\eta_{\text{with}} = \sum_{w=0} k \binom{n}{w} / n. \tag{1}$$

It is unlikely that the load will be perfectly distributed, rather some agents will complete their computations before others and remain idle until all computations are completed. The worst case communicational load per agent is $\mathrm{O}(n^{k-1})$ during the calculation-distribution stage. Alternatively, if each agent is responsible for only computing coalitions in which it is a member, then the number of coalitions evaluated with no communication becomes

$$\eta_{\text{without}} = \sum_{w=0}^{k-1} \binom{n-1}{w}. \tag{2}$$

Equation (2) represents the number of coalitions of size $\leq k$ in which a particular agent A_i is always a member. Equation (1) requires fewer computations than (2) but this is not an order of magnitude difference. The agents' computational load is $\mathrm{O}(n^k)$ per task in both cases. The communication load per robot is $O(1)$ in the calculation-distribution stage. The additional computation may be compensated for by reduced communication time. Experiments from our previous work [29] demonstrated a significant decrease in execution time when communication is removed from the first stage of the algorithm.

A desirable side effect of this modification is additional fault tolerance [28]. If a robot R_A fails during coalition list evaluation, information relevant to coalitions containing R_A is lost. Since coalitions involving R_A cannot be formed post failure, this information is no longer necessary. Thus a robot failure does not require information retrieval from the failed robot. However, the other robots must be aware of the failure so that they can delete all coalitions containing the failed robot R_A.

B. Task Format

Current multi-agent coalition formation algorithms assume that the agents have a capability vector, $\langle b_1^{A_i}, \ldots, b_r^{A_i} \rangle$. Multi-robot capabilities include sensors (i.e., laser range finder or ultrasonic sonars) and actuators (i.e., wheels or gripper). Shehory and Kraus' algorithm assumes that the individual agent resources are collectively available upon coalition formation and that the formed coalition can freely redistribute resources among the software agents. However, this is not possible in a multi-robot domain, as robots cannot autonomously exchange capabilities.

Correct resource distribution is also an issue. The box-pushing task [30] is used to illustrate this point. Three robots, two pushers (with one bumper and one camera each) and one watcher (with one laser range finder and one camera) cooperate to complete the task. The total resource requirements are: two bumpers, three cameras, and one laser range finder. However, this information is incomplete, as it does not represent the constraints related to sensor locations. Correct task execution requires that the laser range finder and camera reside on a single robot while the bumper and laser range finder reside on different robots. This implies that a multi-robot coalition that simply possesses the necessary resources is not necessarily capable of performing a task, the capability locational constraints have to be represented and met.

The constraints can be represented as a Constraint Satisfaction Problem (CSP). The CSP variables are the required sensors and actuators for the task. The domain values for each variable are the available robots possessing the required sensor and actuator capabilities. Two types of constraints exist, the sensors and actuators must reside on the same robot or on different robots. A constraint graph evolves with locational constraints represented as arcs labeled s (same robot) or d (different robot).

Fig. 1 provides the box-pushing task constraint graph. This task's resource constraints between Bumper_1 and Bumper_2 are implied by their locational constraints. Since Bumper_1 and Bumper_2 must be assigned to different robots, there cannot be a solution where a robot with one bumper is assigned to both Bumper_1 and Bumper_2.

The domain values for each variable in Fig. 1 are the robots that possess the capability represented by the variable. A coalition can be verified to satisfy the constraints by applying arc-consistency. If a sensor has an empty domain value set then the current assignment fails and the current coalition is deemed infeasible. A successful assignment indicates the sub-task to which each robot was assigned.

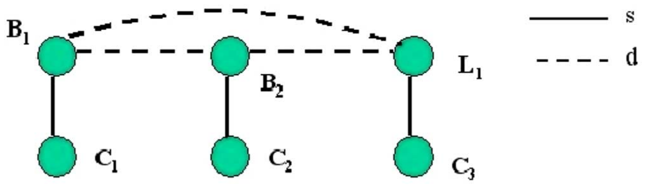

Fig. 1. Box-pushing task constraint graph [29]. (Color version available online at http://ieeexplore.ieee.org.)

Using arc-consistency, each candidate coalition is checked against the constraint graph to verify if its coalition is feasibe. A caveat is that arc-consistency does not detect every possible inconsistency (an NP-complete problem). This limitation may be overcome by solving the CSP for the best coalition selected. If no solution exists (i.e., false positive), then the next best coalition is chosen. Solving the CSP also automatically assigns each robot to the appropriate subtask.

Experiments measuring the additional overhead imposed by the CSP formulation were presented in our earlier work [29]. The experiments demonstrated that the effect of the CSP formulation on the execution time was on the order of milliseconds even for hundreds of agents.

C. Coalition Imbalance

The **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 in which one or more agents have a predominant share of the capabilities may have the same utility as a coalition with evenly distributed capabilities. Robots are 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 as over reliance on dominating members can cause task execution to fail or considerably degrade. If robot R_A is not a dominating member (does not possess many sensors) then it is more likely that another robot with similar capabilities can replace robot R_A.

Rejecting lopsided coalitions in favor of balanced ones is not entirely straightforward. When comparing coalitions of different sizes, a subtle trade-off between lopsidedness and the coalition size can arise. The argument may be made both for fault tolerance and for smaller coalition size. Coalitions with as few robots as possible may be desirable. Conversely, there may be a large number of robots thus placing the priority on fault tolerance and balanced coalitions.

There are some desirable properties for a metric quantifying coalition imbalance. Consider a coalition C with a resource distribution $(r_1, r_2, \ldots, r_n)$ (i.e., coalition member 1 contributes net resources r_1, member 2 contributes net resources r_2, etc.). The chosen balance function should be continuous in r_i, also any change towards a more equable distribution of $r_1, r_2, \ldots, r_n$ should increase the value of the metric. Considering these properties, the **Balance Coefficient** (BC) was introduced to quantify the coalition imbalance level. For coalition C the BC with respect to a particular task can be calculated as follows:

$$\mathrm{BC} = \frac{r_1 \times r_2 \times \dots r_n}{\left[\frac{\text{taskvalue}}{n}\right]^n}. \tag{3}$$

BC measures the deviation from the perfectly balanced coalition where each member contributes equally $(\text{taskvalue}/n)$ to the task. Clearly, the BC is continuous in r_i.

Result: The higher the BC, the more balanced the coalition.

Proof: Consider any coalition of size n with resource distribution $(r_1, r_2, \dots, r_n)$ and assume any task t_l with taskvalue T. Further, consider an integer $s < n$ such that

$$\begin{aligned} r_i &= T/n + \alpha_i, \quad for\ i = 1\ to\ s \\ r_i &= T/n - \delta_i, \quad for\ i = s+1\ to\ n \end{aligned}$$

where $\alpha_i \geq 0$, $\delta_i \geq 0, \forall i$.

The BC for this coalition is

$$\begin{aligned}\gamma \mathrm{BC}_1 = {} & \left(\frac{T}{n} + \alpha_1\right)\left(\frac{T}{n} + \alpha_2\right)\cdots\left(\frac{T}{n} + \alpha_s\right) \\ & \times \left(\frac{T}{n} - \delta_1\right)\left(\frac{T}{n} - \delta_2\right)\cdots\left(\frac{T}{n} - \delta_{n-s}\right). \end{aligned} \tag{4}$$

where $\gamma = (T/n)^n$.

Adding a factor ε to one of the α's and an equal amount μ to all δ's (or *vice versa*) makes the coalition more imbalanced (and should decrease the BC). Thus, the BC becomes

$$\begin{aligned}\gamma \mathrm{BC}_2 = {} & \left(\frac{T}{n} + \alpha_1 + \varepsilon\right)\left(\frac{T}{n} + \alpha_2\right)\cdots\left(\frac{T}{n} + \alpha_s\right) \\ & \times \left(\frac{T}{n} - \delta_1 - \mu_1\right)\cdots\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right) \end{aligned} \tag{5}$$

where

$$\mu_1 + \mu_2 + \mu_{n-s} = \varepsilon, \varepsilon > 0, \mu_i > 0. \tag{6}$$

Factoring out the ε and the μ's, we obtain

$$\begin{aligned}\gamma \mathrm{BC}_2 = {} & \left[\left(\frac{T}{n} + \alpha_1\right)\left(\frac{T}{n} + \alpha_2\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \left. \times \left(\frac{T}{n} - \delta_1\right)\left(\frac{T}{n} - \delta_2\right)\cdots\left(\frac{T}{n} - \delta_{n-s}\right)\right] \\ & + \left[\varepsilon\left(\frac{T}{n} + \alpha_2\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \left. \times \left(\frac{T}{n} - \delta_1 - \mu_1\right)\cdots\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right)\right] \\ & - \left[\mu_1\left(\frac{T}{n} + \alpha_1\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \left. \times \left(\frac{T}{n} - \delta_2 - \mu_2\right)\cdots\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right)\right] \\ & - \left[\mu_2\left(\frac{T}{n} + \alpha_1\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \times \left(\frac{T}{n} - \delta_1\right)\left(\frac{T}{n} - \delta_3 - \mu_3\right)\cdots \\ & \left.\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right)\right]\cdots \\ & - \left[\mu_{n-s}\left(\frac{T}{n} + \alpha_1\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \times \left(\frac{T}{n} - \delta_1\right)\left(\frac{T}{n} - \delta_3\right)\cdots \\ & \left.\left(\frac{T}{n} - \delta_{n-s-1}\right)\right]. \end{aligned} \tag{7}$$

Substituting from (4) into (7)

$$\begin{aligned}\gamma \mathrm{BC}_2 = {} & \gamma \mathrm{BC}_1 + \left[\varepsilon\left(\frac{T}{n} + \alpha_2\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \times \left(\frac{T}{n} - \delta_1 - \mu_1\right)\cdots \\ & \left.\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right)\right] \\ & - \left[\sum_{i=1}^{n-s}\mu_i\left(\frac{T}{n} + \alpha_1\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \times \left(\frac{T}{n} - \delta_1\right)\cdots\left(\frac{T}{n} - \delta_{i-1}\right) \\ & \times \left(\frac{T}{n} - \delta_{i+1} - \mu_{i+1}\right) \\ & \left. \times \left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right)\right]. \end{aligned} \tag{8}$$

Substituting the ε from (6) into (8), we obtain

$$\begin{aligned}\gamma \mathrm{BC}_2 = {} & \gamma \mathrm{BC}_1 + \left[\sum_{i=1}^{n}\mu_i\left(\frac{T}{n} + \alpha_2\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \left. \times \left(\frac{T}{n} - \delta_1\right)\cdots\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right)\right] \\ & - \left[\sum_{i=s+1}^{n-s}\mu_i\left(\frac{T}{n} + \alpha_1\right)\cdots\left(\frac{T}{n} + \alpha_s\right)\right. \\ & \times \left(\frac{T}{n} - \delta_1\right)\cdots\left(\frac{T}{n} - \delta_{i-1}\right) \\ & \times \left(\frac{T}{n} - \delta_{i+1} - \mu_{i+1}\right)\cdots \\ & \left.\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right)\right]. \end{aligned} \tag{9}$$

Separating the common factor results in

$$\begin{aligned}\gamma \mathrm{BC}_2 = {} & \gamma \mathrm{BC}_1 + \sum_{i=1}^{n}\mu_i\left(\frac{T}{n} + \alpha_2\right)\cdots\left(\frac{T}{n} + \alpha_s\right) \\ & \times \left(\frac{T}{n} - \delta_{i+1} - \mu_{i+1}\right)\cdots\left(\frac{T}{n} - \delta_{n-s} - \mu_{n-s}\right) \\ & \times \left[\left(\frac{T}{n} - \delta_1 - \mu_1\right)\cdots\left(\frac{T}{n} - \delta_i - \mu_i\right)\right. \\ & \left. - \left(\frac{T}{n} + \alpha_1\right)\left(\frac{T}{n} - \delta_1\right)\cdots\left(\frac{T}{n} - \delta_{i-1}\right)\right] \end{aligned} \tag{10}$$

now, $((T/n) - \delta_y - \mu_y) < ((T/n) - \delta_y)\ \forall y$.

Therefore, from (10) we obtain

$$\gamma \mathrm{BC}_2 < \gamma \mathrm{BC}_1 + \sum_{i=1}^{n} \mu_i \left(\frac{T}{n}+\alpha_2\right)\cdots\left(\frac{T}{n}+\alpha_s\right) \times \left(\frac{T}{n}-\delta_{i+1}-\mu_{i+1}\right)\cdots\left(\frac{T}{n}-\delta_{n-s}-\mu_{n-s}\right) \times \Bigg[\left(\frac{T}{n}-\delta_1-\mu_1\right)\cdots\left(\frac{T}{n}-\delta_i-\mu_i\right) - \left(\frac{T}{n}+\alpha_1\right)\left(\frac{T}{n}-\delta_1-\mu_1\right)\cdots \left(\frac{T}{n}-\delta_{i-1}-\mu_{i-1}\right)\Bigg]. \tag{11}$$

Separating the common factors a second time results in

$$\gamma \mathrm{BC}_2 < \gamma \mathrm{BC}_1 + \sum_{i=1}^{n} \mu_i \left(\frac{T}{n}+\alpha_2\right)\cdots\left(\frac{T}{n}+\alpha_s\right) \times \left(\frac{T}{n}-\delta_1-\mu_1\right)\cdots\left(\frac{T}{n}-\delta_{n-s}-\mu_{n-s}\right) \times \left[\left(\frac{T}{n}-\delta_i-\mu_i\right)-\left(\frac{T}{n}+\alpha_1\right)\right]. \tag{12}$$

Cancelling equal terms and factoring out the minus sign yields

$$\Rightarrow \gamma \mathrm{BC}_2 < \gamma \mathrm{BC}_1 - \sum_{i=1}^{n} \mu_i \left(\frac{T}{n}+\alpha_2\right)\cdots\left(\frac{T}{n}+\alpha_s\right) \times \left(\frac{T}{n}-\delta_1-\mu_1\right)\cdots\left(\frac{T}{n}-\delta_{n-s}-\mu_{n-s}\right) \times (\delta_i+\mu_i+\alpha_1). \tag{13}$$

$$\Rightarrow \gamma \mathrm{BC}_2 < \gamma \mathrm{BC}_1 - \Psi. \tag{14}$$

where

$$\Psi = \sum_{i=1}^{n} \mu_i \left(\frac{T}{n}+\alpha_2\right)\cdots\left(\frac{T}{n}-\alpha_s\right) \times \left(\frac{T}{n}-\delta_1-\mu_1\right)\cdots\left(\frac{T}{n}-\delta_{n-s}-\mu_{n-s}\right) \times (\delta_i+\mu_i+\alpha_1) > 0. \tag{15}$$

Hence, $\mathrm{BC}_2 < \mathrm{BC}_1$. ◇ (16)

The proof for the case when we increase one of the α's and multiple δ's is similar.

Result: The BC for a perfectly balanced coalition is always 1.

Proof: Consider any coalition with n members, each contributing equally to the utility coalition (i.e., the resource distribution is $r_1, r_2, \ldots, r_n$ where $r_1 = r_2 = r_3 = \ldots = r_n = \text{taskvalue}/n$). From (3), the BC is given by

$$\mathrm{BC} = \frac{r_1 \times r_2 \ldots \times r_n}{(\text{taskvalue}/n)^n} = \frac{(\text{taskvalue}/n)^n}{(\text{taskvalue}/n)^n} = 1. \tag{17}$$

Therefore, a perfectly balanced coalition always has a $\mathrm{BC} = 1$.◇

Corollary: The value of the BC can never exceed 1.

Proof: Since a perfectly balanced coalition has a BC of 1, and the BC increases with the level of balance, then no coalition can have a BC in excess of a perfectly balanced coalition. ◇

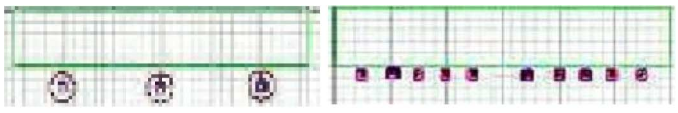

Fig. 2. Two perfectly balanced coalitions of different sizes. (a) Three robot coalition. (b) Ten robot coalition. (Color version available online at http://ieeexplore.ieee.org.)

The BC is useful for comparing the level of imbalance across coalitions of the same size. However, the BC alone may not permit comparison of variable sized coalitions from a fault tolerance perspective. For example, Fig. 2 shows two perfectly balanced coalitions performing the same box-pushing task. Fig. 2(a) shows a coalition comprised of three large, more capable robots and Fig. 2(b) shows a coalition comprised of ten small, less capable robots. The BC for both coalitions is 1, thus the BC alone cannot discrimate between two differently sized coalitions. Generally, larger coalitions imply that the average individual contribution and the capability requirements from each member is lower. Thus the comparison across coalitions of different sizes requires that the metric subsume elements of both balance and size. The **Fault Tolerance Coefficient** (FTC) is such a metric and has the following form:

$$\mathrm{FTC} = w_1 \times (\text{balance metric}) + w_2 \times (\text{size function}) \tag{18}$$

where $w_1 + w_2 = 1.0$.

The size function may be a monotonically increasing or a decreasing function of coalition size (n) depending on whether the user favors smaller or larger coalitons. The weights may be adjusted in according to the importance of both the balance and the number of robots. The following size function is utilized in this work:

$$f(n) = 1 - e^{-\lambda n}, \quad 0 < \lambda < 1. \tag{19}$$

The function in (19) is montonic and asymptotically approaches 1 (like the BC, it never exceeds 1; see Fig. 3). Another important property is that after a particular point, increasing n does not result in a significant increase to the function value, i.e., the function converges to 1. This is desirable from a coalition formation perspective since after a certain point, increasing coalition size does not yield improved performance. The exact size at which the function stabilizes can be altered by varying λ in (19). The balance metric in (18) is any appropriate metric that satisfies the properties mentioned earlier. The balance coefficient (3) is the chosen balance metric in this paper.

The question is how to incorporate the FTC into the algorithm in order to select better coalitions. Initially the algorithm proceeds as in Section III, determining the best-valued coalition without considering lopsidedness. As a modification, a list of all coalitions is maintained whose values are within a certain range (5%) of the best coalition value. The modified algorithm then calculates the FTC for all these coalitions and chooses the

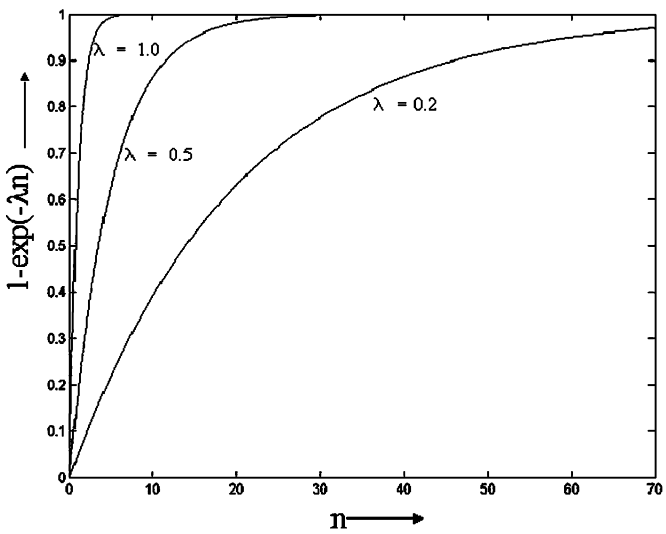

Fig. 3. Size function with λ values of 0.2, 0.5, and 1.0 as size increases.

one with the highest FTC. This ensures that if there is a coalition whose value is within a bound of the highest coalition value and is more fault tolerant, then the algorithm favors the coalition with higher FTC.

D. Further Optimizations

The algorithm complexity can be significantly reduced if the robots are classified according to capability requirements. For example, if the number of identical robots exceeds the maximum coalition size k, then the number of robots in that category can be assumed to be equal to k. If there are 100 available robots, each with one camera and one laser range finder, and the maximum coalition size is ten, then the coalition enumeration can assume that there are only ten identical robots of this type. Since the robots of a particular type are identical, coalitions with up to ten robots may be composed of any of the 100 robots within that type. Hence, the number of candidate coalitions drops from $\approx 100^{10}$ to 10.

E. The Multi-Robot Coalition Formation Algorithm

The coalition formation algorithm is iterative and a task is allocated at each iteration. Within each iteration, the algorithm proceeds in two stages:

1) All possible coalitions are distributively calculated and the initial coalition values are computed.
2) Agents agree on the preferred coalitions and form them.

Stage 1-Preliminary Coalition Evaluation: Initially each agent A_i has a list of agents A and a list of coalitions C_{list} in which A_i is a member. A_i performs the following steps for each coalition C in C_{list}:

1) Calculate the coalitional capabilites vector (B_c), by summing the capabilities of the coalition members. Formally, $B_C = \sum_{A_i \in C} B_{A_i}$.
2) Form a list (E_c) of the expected outcomes of the tasks in set TS when coalition C performs those tasks. For each task $t_j \in TS$:
   a) Determine the necessary capabilities for task t_j.
   b) Compare t_j's capability vector B_j to the sum of the coalition capabilities B_c.
   c) If $\forall\ i,\ b_i^{t_j} \leq b_i^C$ then utilize the CSP formulation (Section IV-B) to verify the locational sensor constraints for the coalition members.
   d) If the constraints are met, then calculate t_j's expected net outcome (e_j) with respect to C by subtracting the cost of unutilized resources from the net task-value. This is the expected net outcome (e_j) of the coalition-task pair $\langle C, t_j \rangle$. Place $\langle e_j, C, t_j \rangle$ into E_c.
   e) Choose the highest valued coalition-task pair from E_c and place it in H_{CT}.

At the end of Stage 1, each agent has a list of coalition-task pairs and coalition values.

Stage 2-Final Coalition Formation: Each agent (A_i) iteratively performs the following:

1) Locate, in H_{CT} the coalition-task pair $\langle C_{\max}, t_{\max} \rangle$ with the highest value $e_{\max}$.
2) Retain in H_{CT} all coalition-task pairs with values within a bound (5%) of $e_{\max}$.
3) Calculate the BC for all coalition-task pairs in H_{CT}.
4) Broadcast the coalition-task pair with the highest BC, $\langle C_{\text{BC}}, t_{\text{BC}} \rangle$ along with the coalition value e_{BC}.
5) Choose the coalition, task pair $\langle C_{\text{high}}, t_{\text{high}} \rangle$ with the highest value e_{high} from all broadcasted coalition pairs.
6) If A_i is a member of coalition C_{high}, join C_{high}, return.
7) Delete from C_{list} coalitions containing members of C_{high}.
8) Delete the chosen task t_{high} from TS.

The above steps are repeated until all the agents are deleted, until there are no more tasks to allocate, or none of the possible coalitions is beneficial. The complexity of this algorithm is unchanged from that of the multi-agent algorithm [3]. The only additional overhead is due to the application of arc-consistency for constraint checking (Section IV). Arc-consistency runs in $O(q^2k^3)$ time per coalition where q is the maximum number of capabilities required for a task and k is the maximum coalition size. Since both q and k do not depend on the total number of robots or the number of tasks, the check requires $O(1)$ operations. Therefore, choosing the largest valued coalition is on the order of the number of coalitions, i.e., $O(n^{k-1})$ [3]. Thus, the CSP formulation does not alter the complexity of the algorithm. An empirical evaluation of the effect of the CSP formulation on the running time of the algorithm is provided in our previous work [29].

V. Experiments

Preliminary experiments [29] demonstrated the validity of the suggested algorithm modifications. Four additional experiments were conducted to validate the algorithm with a number of robots. The first two experiments demonstrate the impact of the FTC on the resulting coalitions, both in simulation and with real world robots. The final two experiments demonstrate the algorithm's applicability to real world tasks in the multi-robot domain. Videos of the experiments are available at: http://www.vuse.vanderbilt.edu/~adamsja/Laboratory/Media.html

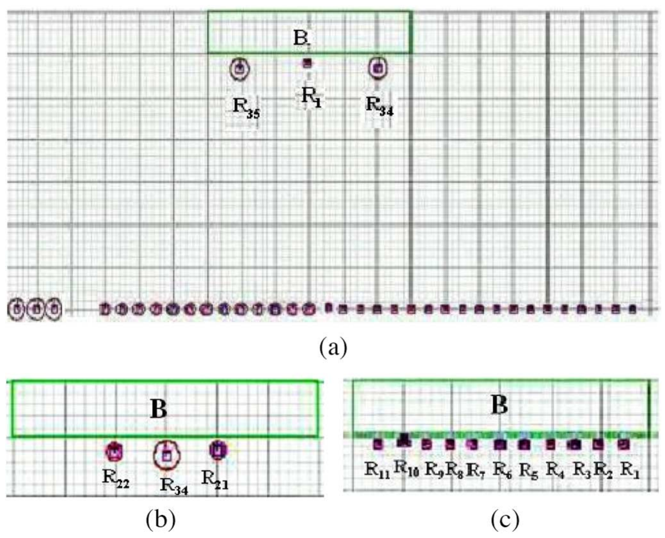
B
R_{35}
R_1
R_{34}
(a)
B
R_{22}
R_{34}
R_{21}
(b)
B
R_{11} R_{10} R_9 R_8 R_7 R_6 R_5 R_4 R_3 R_2 R_1
(c)

Fig. 4. The coalition formed (a) without the FTC, (b) with the FTC and a size function $= -f(n)$, (c) with FTC and a size function $= f(n)$. (Color version available online at http://ieeexplore.ieee.org.)

A. *Fault Tolerance Coefficient Experiment*

This experiment demonstrates the effect of utilizing the FTC to favor the creation of more fault tolerant coalitions. The Player/Stage simulation environment [31] was employed. The tasks required pushing a very large box by jointly exerting forces on the box. The degree of task difficulty was adjusted by varying the box's size and its coefficient of friction with the floor. Adjusting the forces the robots could exert varied the robots' capabilities. (Note: boxes in the simulations did not actually move.) The FTC was

$$\text{FTC} = w_1 \times \text{BC} + w_2 \times [f(n)], \quad w_1 = w_2 = 0.5 \qquad (20)$$

where

$$f(n) = [1 - exp(-\lambda n)], \quad \text{with } \lambda = 0.5. \qquad (21)$$

Box-pushing required the robot to possess a laser range finder, be mobile, be able to exert a certain force F, and be able to communicate with coaliton members. Forty simulated robots were employed, as shown in Fig. 4(a). The robots were numbered 1 to 40 from right to left along the bottom of the figure. Each robot had a specific force capability type: small robots exerted five units of force (the rightmost 20 robots), medium sized robots exerted 15 units of force (the middle 13 robots) and large robots had 25 units of force (the leftmost seven robots). The robots used the incremental SLAM algorithm [31] for localization and the vector field histogram algorithm [32] for navigation and obstacle avoidance. The maximum allowed coalition size (k) was fixed at 15.

Simulation snapshots are provided for a task requiring 55 units of force. Fig. 4(a) shows the resulting coalition without incorporating fault tolerance. The coalition is comprised of two large robots (R_{34}, R_{35}) and one small robot (R_1). The BC and FTC values for this coalition are 0.51 and 0.60, respectively.

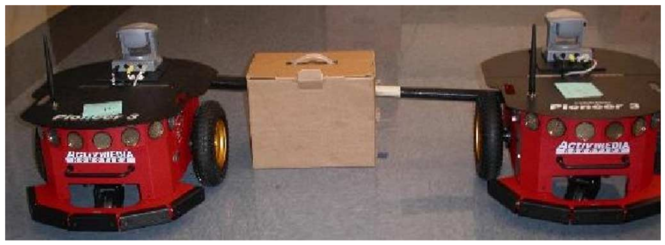

Fig. 5. Two pioneer DX robots pushing a box after forming a coalition. (Color version available online at http://ieeexplore.ieee.org.)

Fig. 4(b) shows the same task performed while incorporating the FTC with a decreasing size function $-f(n)$, placing a low priority on fault tolerance and a high priority on minimizing the number of robots. The coalition is comprised of two medium sized robots (R_{21}, R_{22}) and one large robot (R_{34}). The resulting coalition is more balanced and has a higher BC (0.91) and consequently a higher FTC (0.80) than the FTC value of the coalition in Fig. 4(a).

Fig. 4(c) depicts the experiment conducted with size function, $f(n)$, which favors the formation of larger coalitions. The resulting coalition consists of 11 small robots $(R_1, R_2, \ldots, R_{11})$. Thus, a perfectly balanced coalition is obtained. The advantage is that a larger number of small, less capable robots should have higher fault tolerance. If one robot fails, it should be easier to replace as opposed to replacing a larger, more capable robot. The coalition's FTC for this coalition is the highest of all possible coalitions [Fig. 4(a)–(c)] at 0.996.

B. *Real World Experiments*

The FTC simulation experiments were ported to real robots. The challenge was to find suitable tasks that the robots could perform and whose difficulty could be varied, while also quantifying the robots' capabilities so that a robot's utility for a particular task could be assessed.

The experiments ported the algorithm to three Pioneer 3-DX robots. The experimental tasks involve pushing box through a distance of one meter in a straight line from its initial position. A rod was inserted through the box, preventing interference between the robots. The maximum coalition size is restricted to two robots. The robots position themselves so that the box's net torque is approximately zero in order to ensure that the box does not rotate beyond the acceptable limits. The FTC parameters were identical to those defined for the experiments in Section V-A, where the size function is as defined in (21). Since the size is constant for these experiments, the FTC is equivalent to the BC.

Velcro was attached to the rod to prevent slippage. The boxes contained weights that permitted variation in task difficulty. The robots know their initial positions, the box position, and navigate using odometry. The robots' capabilities were changed based on varying the robots' speed. Every 0.05 m/s of speed increases the robots' capability by ten units. Each pound of weight in the box increased the task value by ten. The purpose was to verify the task allocation and coalition formation rather than monitoring the quality of the task execution. Fig. 5 shows the experimental execution.

TABLE I
COALITION FORMATION RESULTS FOR REAL BOX-PUSHING TASKS

Reading	Robot Capabilities			Task Value		Coalitions Formed			
	Robot1	Robot2	Robot3	Task1	Task2	With *BC*		Without *BC*	
						Members	*BC*	Members	*BC*
1	40	40	60	80	100	1,2	1	1,3	0.960
2	20	20	30	40	50	1,2	1	1,3	0.960
3	45	25	25	50	70	2,3	1	1,2	0.968
4	25	20	20	50	70	-	-	-	-
5	15	20	10	50	60	-	-	-	-
6	35	35	40	100	80	-	-	-	-
7	10	15	15	20	30	2,3	1	2,3	1
8	20	25	25	70	50	2,3	1	2,3	1
9	30	30	30	50	60	1,2	1	1,2	1

The experimental results are shown in Table I. Size did not play a role in determining the coalitions that were formed since only three robots were employed. The coalition formation was influenced primarily by the BC. Readings 1–3 demonstrate that different robotic coalitions form depending on whether or not the BC is utilized. For example, in Reading 1, robots 1 and 2 form a coalition to complete task 1 even though there is the potential for another coalition ([2, 3] or [1, 3]) to perform the more valuable task 2. This is because the BC is higher for the task 1 coalition [1,2] than the for the potential task 2 coalitions [1, 3] or [2, 3].

Readings 4–6 demonstrate situations where the three robots could form a coalition but do not because the maximum coalition size is limited to two robots. This result demonstrates that the algorithm does not perform an exhaustive search through the coalition space, rather it only considers coalitions with fewer than k (here two) members.

The final readings 7–9 are indicative of situations when the BC has no impact on the resulting coalitions. The BC value is inconsequential because the highest valued coalition also has BC $= 1$.

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

C. Coalition Formation Experiments

The experiments in Section V-A and V-B provided preliminary verification of the algorithm's applicability to real world tasks, however, it was necessary to validate the algorithm with a larger number of robots. Experiments were conducted with simulated and real world robots that had heterogeneous sensory capabilities. Each robot possessed a subset of the following capabilities: bumper, laser range finder, camera, mobile, communications, and ultrasonic sonar. The FTC employed in these experiments is identical to that used in Section V-B. The following tasks were employed for the experiments:

- **Box-Pushing**: This task requires two robots to form a coalition, navigate through the environment from their initial locations to either side of a large box and simultaneously push the box through a straight distance of one meter. The task requires that the two robots be equipped with laser range finders, be able to communicate with each other, and be mobile. The robots navigate through the environment using the vector field histogram algorithm [32] and orient themselves perpendicular to the box using the laser range finders. The incremental SLAM algorithm [31] provided laser corrected odometery. A synchronizing message ensured that the robots pushed the box simultaneously. This task requires the coalition members perform their actions in a synchronized fashion and hence is a tightly-coupled task.
- **Cleanup**: This task involves two to five robots forming a coalition to clear a room of small colored boxes by pushing them to the edge of the room. Each robot requires a camera, a set of sonars, and must be mobile. A behavior based approach was employed to perform the tasks. Behaviors include locating the colored boxes and pushing them towards the wall. The robots use a sonar based obstacle avoidance behavior and the cameras for object (box) tracking. This task does not require synchronization between robots and therefore is a loosely-coupled tasks.
- **Sentry Duty**: This task involves two to four robots forming a coalition to monitor different areas of the environment for motion. The task requires that the robots be equipped with laser range finders and be mobile. Again the vector field histogram algorithm is used for navigation and the laser range finders are used to detect motion. The incremental SLAM algorithm provided laser corrected odometery. The robots do not communicate with each other, but the coalition members combine to monitor a particular area for motion. Therefore, this task falls in between a tightly-coupled and a loosely-coupled task.

1) Single Task Robot Validation: These experiments demonstrated that the coalition formation algorithm operates independent of the nature of the tasks and the task methodology utilized to perform those tasks. Separate experiments were performed with the box-pushing, sentry-duty, and cleanup tasks. The simulation and real world experiments for each type of task are shown in Table II. The maximum coalition size for all experiments was four. All real-world experiments, involved a total of 12 heterogeneous Pioneer DX robots.

The box-pushing task required two robots form a coalition to push a large box through a straight distance of one meter. The tightly coordinated task required synchronous message passing between robots. Experiment 1, in Table II, represents a simulation run with four box-pushing tasks and ten simulated robots.

TABLE II
RESULTS FROM SIMULATED AND REAL ROBOTS FORMING COALITIONS FOR A SINGLE TASK

Reading	Task Type	Task Methodology	Simulation		Real World Robots	
			N_{Robots}	N_{Tasks}	N_{Robots}	N_{Tasks}
1	Box-pushing	Synchronized (*Tightly coupled*)	10	4	12	2
2	Foraging	Behavior-Based (*Loosely coupled*)	12	1	12	1
3	Sentry-Duty	Semi-Synchronized (*Intermediate coupling*)	9	2	12	2

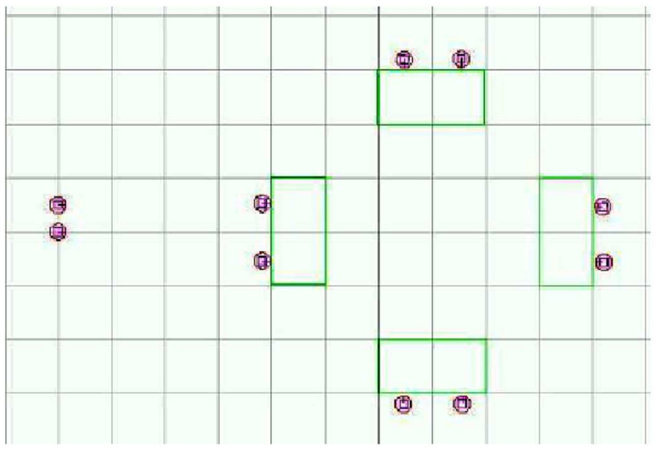
Fig. 6. Simulated coalitions of two robots performing four box-pushing tasks. (Color version available online at http://ieeexplore.ieee.org.)

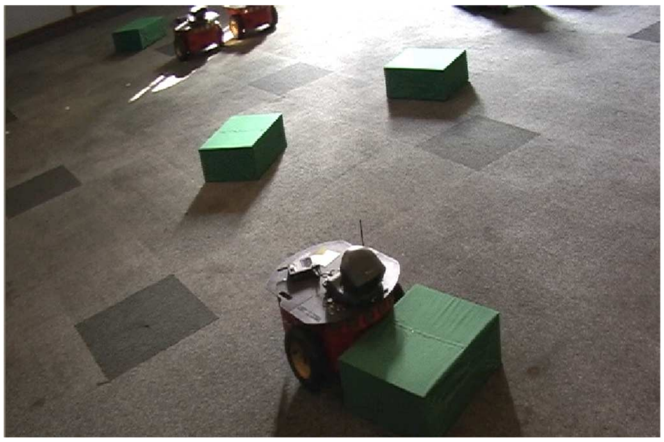
Fig. 8. Robot coalition performing the real-world cleanup task. (Color version available online at http://ieeexplore.ieee.org.)

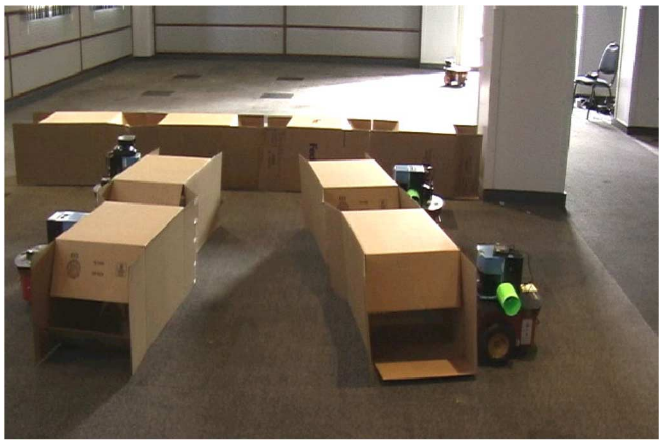
Fig. 7. Two coalitions of two robots performing two box-pushing tasks. (Color version available online at http://ieeexplore.ieee.org.)

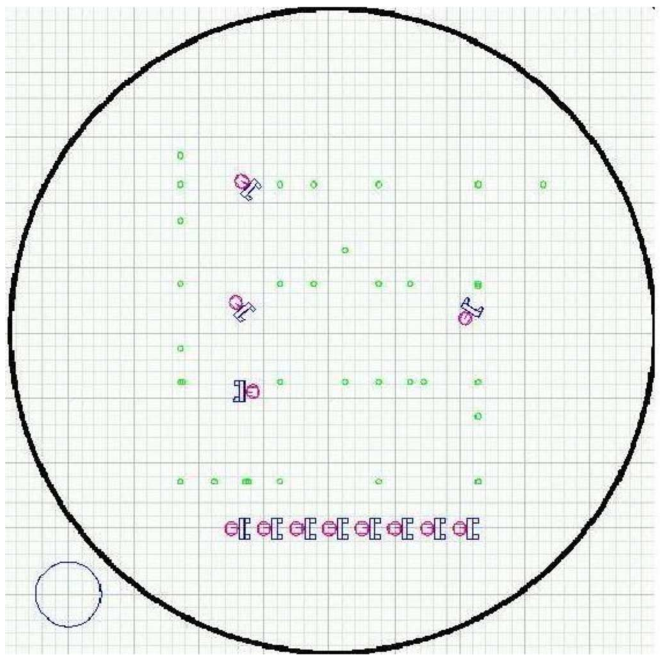
Fig. 9. Simulated four robot coalition performing a foraging task. (Color version available online at http://ieeexplore.ieee.org.)

All four box-pushing tasks were successfully allocated and performed by four different coalitions, as shown in Fig. 6. The corresponding real world experiment had two box-pushing tasks, each requiring two robots that were successfully allocated; see Fig. 7.

The cleanup task (see Fig. 8) involved forming a coalition to clear an area of boxes. The implementation is behavior based with robots performing a random walk until a colored object is identified via image processing. The robot then moved towards the object (and consequently pushed it) until its sonar sensors indicated that the robot was close to a wall, in which case the object was deposited, obstacle avoidance became active, the robot turned, and resumed a random walk. The corresponding simulation task is the foraging task that involved robots picking up objects of interest (pucks) and depositing them at a goal location. Experiment 2, in Table II, represents a simulation run requiring four robots to form a coalition from twelve simulated robots. Fig. 9 shows four robots performing the foraging task.

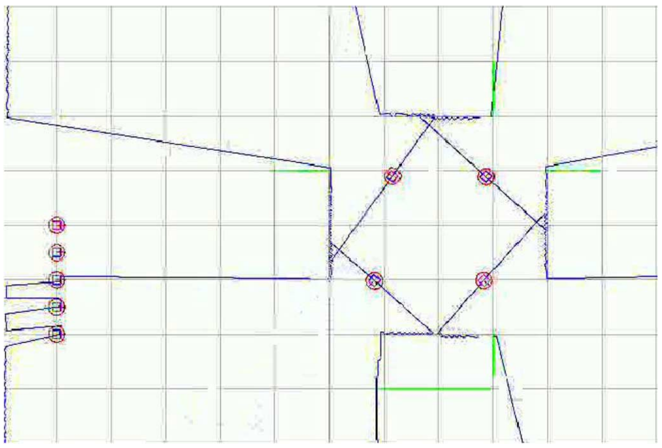

Fig. 10. Two coalitions (two robots each) performing a sentry-duty task. (Color version available online at http://ieeexplore.ieee.org.)

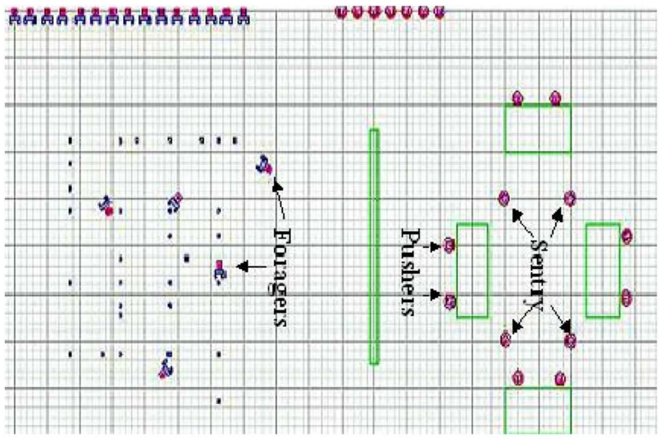

Fig. 11. Simulation of four two-robot box-pushing tasks, two two-robot sentry-duty tasks, and a four robot foraging task. (Color version available online at http://ieeexplore.ieee.org.)

The corresponding real world cleanup task involved four robots forming a coalition.

The sentry-duty task involved a team navigating through the environment to fixed positions (normally an entrance or exit) and performing motion detection from these positions. Fig. 10 shows a simulation run in which two coalitions were formed comprised of two robots each from a possible nine robots. This simulation is represented by Experiment 3 in Table II. During the real robot sentry-duty experiments, two tasks were successfully allocated, each requiring two robots.

All tasks were successfully allocated and performed by the coalition formation algorithm in both simulation and with real robots. These results demonstrate that the algorithm operates independently of the nature of the tasks (loosely-coupled versus tightly-coupled), or the methodology utilized to perform the tasks (behavior-based versus synchronized).

2) Multiple Task Robot Validation: These experiments demonstrated that the task allocation operates independently of task diversity. Separate coalitions were formed for combinations of the box-pushing (tightly-coupled) task, the cleanup task (loosely-coupled, behavior-based), and the sentry-duty task (intermediate-coupling). Fig. 11 shows a simulation that

TABLE III
RESULTS FROM SIMULATED ROBOTS FORMING COALITIONS FOR MULTIPLE TASKS

Reading	box-pushing		foraging		sentry-duty	
	N_{tasks}	N_{alloc}	N_{tasks}	N_{alloc}	N_{tasks}	N_{alloc}
1	4	4	1	1	2	2
2	3	3	2	2	2	2
3	4	4	2	2	2	2

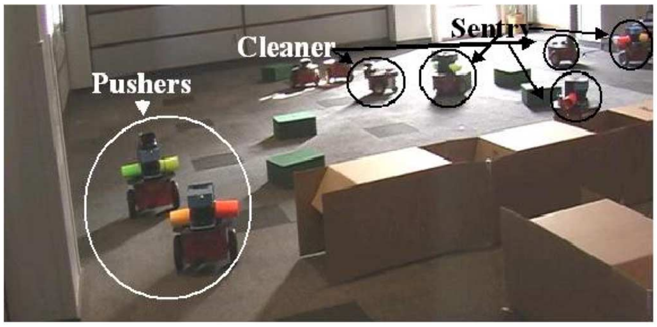

Fig. 12. The real robots performing a combination of box-pushing, sentry-duty and cleanup tasks. (Color version available online at http://ieeexplore.ieee.org.)

involved four distinct box-pushing tasks, a foraging task, and two sentry-duty tasks. The coalitions were chosen from 40 heterogeneous robots. Table III illustrates three simulation runs with different combinations of each of the three tasks. All experiments involved combinations of the box-pushing task requiring two robots, the cleanup task requiring four robots, and the sentry-duty task requiring two robots. Experiment 1 represents a simulation with four box-pushing tasks, one foraging task, and two sentry-duty tasks. Experiment 2 represents a simulation with three box-pushing tasks, two foraging tasks, and two sentry-duty tasks. Experiment 3 represents a simulation with four box-pushing tasks, two foraging tasks, and two sentry-duty tasks. All three runs represent simulations where the algorithm successfully allocated all tasks.

The real world robot experiments employed 13 robots. Eight robots were equipped with laser range finders and five were equipped with cameras. All real robot tasks required the same number of robots as defined for the simulation experiments. Fig. 12 shows an experiment with a cleanup task, sentry-duty task, and a box-pushing task. Table IV illustrates experiments with different combinations of the three tasks performed by the real robots. Experiment 1 represents a situation in which the task requirements are met and all tasks are successfully allocated. Experiment 2 represents a situation in which all tasks could not be successfully allocated because of insufficient resources. Experiment 3 represents a situation where the resources exceed the resource requirements. Both Tables III and IV demonstrate that the algorithm is applicable to a combination of task types (and task methodologies).

Each case allocated the tasks to coalitions that successfully performed the task. Fig. 13 indicates the message traffic rate for a robot participating in the task allocation process. The peaks correspond to the broadcasting of coalition-task values, while the flat regions correspond to periods spent evaluating the coalition lists. As more tasks are allocated, fewer robots remain, and

TABLE IV
RESULTS FROM REAL WORLD ROBOTS FORMING COALITIONS FOR MULTIPLE TASKS

Reading	box-pushing		cleanup		sentry-duty	
	N_{tasks}	N_{alloc}	N_{tasks}	N_{alloc}	N_{tasks}	N_{alloc}
1	2	2	1	1	2	2
2	2	2	4	1	10	4
3	1	1	1	1	2	2

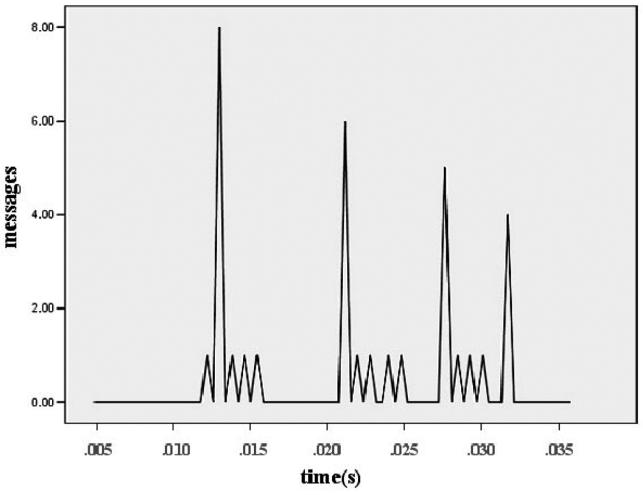

Fig. 13. Messaging traffic as time progresses and the number of robots participating decreases.

the number of transmitted messages decreases with each iteration. The sharpest spike or messaging burst required a bandwidth of approximately 2.34 Kbps, which is very acceptable given the available bandwidth on modern networks. This suggests that the algorithm's messaging requirements should scale nicely for even larger numbers of robots.

D. Discussion

The level of imbalance has important implications with regard to a coalition's level of fault tolerance (Section IV-C). The effect of incorporating the FTC on the coalition formation was demonstrated in Sections V-A and V-B. Three different tasks were defined and tested both in simulation and with real robots in order to validate the algorithm with a large number of robots. The tasks required different levels of coupling and each task required a different methodology for task execution. The results in Section V-C.1 demonstrate that the algorithm operates independent of the nature of the tasks or task methodology. The experiments in Section V-C.2 demonstrate that the algorithm is able to simultaneously allocate different types of tasks.

VI. CONCLUSION

Finding the optimal multi-robot coalition for a task is an intractable problem. This work shows that, with certain modifications, coalition formation algorithms provided in the multi-agent domain can be applied to the multi-robot domain. The concept of coalition imbalance was introduced and its impact on the coalition's fault tolerance was demonstrated. Metrics for measuring balance and the fault tolerance of a coalition were provided. Finally, the algorithm was demonstrated to work in simulation and on a set of Pioneer-DX robots.

ACKNOWLEDGMENT

The authors thank Dr. P. Molnar for access to the robotics laboratory at Clark Atlanta University. The authors also thank Dr. S. Hutchinson, Dr. L. E. Parker, and the anonymous reviewers for their valuable feedback.

REFERENCES

[1] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tomhe, "Coalition structure generation with worst case guarantees," *Artif. Intell.*, vol. 111, no. 1–2, pp. 209–238, 1999.
[2] M. Klusch and O. Shehory, "A polynomial kernel-oriented coalition formation algorithm for rational information agents," in *Proc. Int. Conf. MultiAgent Syst.*, 1996, pp. 157–164.
[3] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell. J.*, vol. 101, no. 1–2, pp. 165–200, 1998.
[4] ——, "Feasable formation of coalitions among autonomous agents in non-super-additive environments," *Comput. Intell.*, vol. 15, no. 3, pp. 218–251, 1999.
[5] B. Gerkey and M. J. Matarić, "A framework for studying multi-robot task allocation," in *Proc. Multi-Robot Syst.: From Swarms to Intell. Automata*, 2003, vol. 2, pp. 15–26.
[6] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multi-robot cooperation," *IEEE Trans. Robot. Autom.*, vol. 14, no. 2, pp. 220–240, Apr. 1998.
[7] B. P. Gerkey and M. J. Matarić, "MURDOCH: Publish/subscribe task allocation for heterogeneous agents," in *Proc. Auton. Agents*, 2000, pp. 203–204.
[8] B. Werger and M. J. Matarić, "Broadcast of local eligibility: Behavior based control for strongly cooperative multi-robot teams," in *Proc. Auton. Agents*, 2000, pp. 21–22.
[9] M. B. Dias, "Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments," Ph.D. dissertation, Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, 2004.
[10] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.
[11] E. Balas and M. Padberg, "Set partitioning: A survey," *SIAM Rev.*, vol. 18, pp. 710–760, 1976.
[12] P. C. Chu and J. E. Beasley, "A genetic algorithm for the set covering problem," *Eur. J. Oper. Res.*, vol. 94, pp. 396–404, 1996.
[13] M. Fisher and P. Kedia, "Optimal solution of set covering/partitioning problems using dual heuristics," *Manage. Sci.*, vol. 36, no. 6, pp. 674–688, 1990.
[14] K. Hoffman and M. Padberg, "Solving airline crew-scheduling problems by branch-and-cut," *Manage. Sci.*, vol. 39, no. 6, pp. 667–682, 1993.
[15] L. S. Shapely and M. Shubik, *Game Theory in Economics*. Santa Monica, CA: Rand, 1973.
[16] S. Kraus, J. Wilkenfeld, and G. Zlotkin, "Multiagent negotiation under time constraints," *Artif. Intell.*, vol. 75, no. 2, pp. 297–345, 1995.
[17] R. G. Smith, "The contract net protocol: High level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, 1980.
[18] T. W. Sandholm and V. R. Lesser, "Coalition formation among bounded rational agents," in *Proc. Int. Joint Conf. Artif. Intell.*, 1995, pp. 662–669.
[19] R. Zlot and A. Stentz, "Complex task allocation for multiple robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 67–72.
[20] F. Tang and L. E. Parker, "ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 1770–1777.
[21] B. R. Donald, J. Jennings, and D. Rus, "Information invariants for distributed manipulation," *Int. J. Robot. Res.*, vol. 16, no. 5, pp. 673–702, 1997.
[22] F. Tang and L. E. Parker, "Coalescing multi-robot teams through ASyMTRe: A formal analysis," in *Proc. IEEE Int. Conf. Adv. Robot.*, 2005, pp. 817–824.

[23] V. D. Dang and N. Jennings, "Generating coalition structures with finite bound from the optimal guarantees," in *Proc. 3rd Int. Joint Conf. Auton. Agents and MultiAgent Syst.*, 2004, pp. 564–571.

[24] L. F. Fass, "Automatic-theoretic view of agent coalitions," Amer. Assoc. Artif. Intell. Tech. Rep. WS-04-06, 2004, pp. 18–21.

[25] X. Li and L.-K. Soh, "Investigating reinforcement learning in multiagent coalition formation," Amer. Assoc. Artif. Intell. Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems Tech. Rep. WS-04-06, 2004, pp. 22–28.

[26] R. Sorbella, A. Chella, and R. Arkin, "Metaphor of politics: A mechanism of coalition formation," Amer. Assoc. Artif. Intell. Tech. Rep. WS-04-06, 2004, pp. 45–53.

[27] S. Abdallah and V. Lesser, "Organization-based cooperative coalition formation," in *Proc. IEEE/WIC/ACM Int. Conf. Intell. Agent Technol.*, 2004, pp. 162–168.

[28] P. Ulam and R. Arkin, "When good comms go bad: Communications recovery for multi-robot teams," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, vol. 4, pp. 3727–3734.

[29] L. Vig and J. A. Adams, "Issues in multi-robot coalition formation," in *Proc. Multi-Robot Syst. From Swarms to Intell. Automata*, 2005, vol. 3, pp. 15–26.

[30] B. P. Gerkey and M. J. Matarić, "Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 464–469.

[31] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, G. S. Sukhatme, and M. J. Matarić, "Most valuable player: A robot device server for distributed control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2001, pp. 1226–1231.

[32] J. Borenstein and Y. Koren, "The vector field histogram—Fast obstacle-avoidance for mobile robots," *IEEE J. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.

Lovekesh Vig received the B.S. degree in physics and M.S. in computer science from Delhi University, Delhi, India, in 1999 and 2002, respectively. He is currently working toward the Ph.D. degree in computer science at Vanderbilt University, Nashville, TN.

His research interests include distributed algorithms and architectures for multiple-robot cooperation and coordination.

Julie A. Adams (M'91–SM'01) received the B.S. degree in computer science and the B.B.A. degree in accounting from Siena College, Loudonville, NY, in 1989 and 1990, respectively, and the M.S.E and Ph.D. degrees in computer and information sciences from the University of Pennsylvania, Philadelphia, in 1993 and 1995, respectively.

She worked in Human Factors for Honeywell, Inc. and the Eastman Kodak Company between 1995 and 2000. She joined the Department of Computer Science, Rochester Institute of Technology, Rochester, NY, in 2000. She conducts research in human–robotic interaction and distributed algorithms for multiple robotic systems.

Dr. Adams has held various positions within the IEEE Systems, Man, and Cybernetics Society and has served on the IEEE Technical Activities Board Finance and Society Review Committees. She is a member of the Human Factors and Ergonomics Society, Association of Computing Machinery, and the American Association of Artificial Intelligence.