

ISSUES IN MULTI-ROBOT COALITION FORMATION

Lovekesh Vig

Electrical Engineering and Computer Science Department

Vanderbilt University, Nashville TN 37212

lovekesh.vig@vanderbilt.edu

Julie.A.Adams

Electrical Engineering and Computer Science Department

Vanderbilt University, Nashville TN 37212

julie.a.adams@vanderbilt.edu

Abstract Numerous coalition formation algorithms exist in the Distributed Artificial Intelligence literature. Algorithms exist that form agent coalitions in both super additive and non-super additive environments. The employed 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 the game theory literature.

Despite the plethora of multi-agent coalition formation literature, to the best of our knowledge none of these algorithms have been demonstrated with an actual multiple-robot system. There exists a discrepancy between the multi-agent algorithms and their applicability to the multiple-robot domain. This work aims to correct that discrepancy by unearthing issues that arise while attempting to tailor these algorithms to the multiple-robot domain. A well-known multiple-agent coalition formation algorithm has been studied in order to identify the necessary modifications to facilitate its application to the multiple-robot domain.

Keywords: Coalition formation, fault-tolerance, multi-robot, task allocation.

1. Introduction

Multi-agent systems often encounter situations that require agents to cooperate and perform a task. In such situations it is often beneficial to assign a group of agents to a task, such as when a single agent cannot

perform the tasks. This paper investigates allocating tasks to disjoint robot teams, referred to as coalitions. Choosing the optimal coalition from all possible coalitions is an intractable problem due to the size of coalition structure space (Sandholm et al., 1999). Algorithms exist that yield solutions within a bound from the optimal and are tractable. However these algorithms make underlying assumptions that are not applicable to the multiple-robot domain, hence the existence of a discrepancy between the multi-agent and multiple-robot coalition formation literature. This paper identifies these assumptions and provides modifications to the multi-agent coalition formation algorithms to facilitate their application in the multiple-robot domain. Gerkey and Mataric (Gerkey and Mataric, 2004) indicate that despite the existence of various multi-agent coalition formation algorithms, none of these algorithms have been demonstrated in the multiple-robot domain.

Various task allocation schemes exist. The ALLIANCE (Parker, 1998) architecture uses motivational behaviors to monitor task progress and dynamically reallocate tasks. The MURDOCH (Gerkey and Mataric, 2002) and BLE (Werger and Mataric, 2000) systems use a Publish/Subscribe method to allocate tasks that are hierarchically distributed. However, most current task allocation schemes assume that all of the system robots are available for task execution. These systems also assume that communication between robots is always possible or that the system can provide motivational feedback. These assumptions need not always hold, a set of tasks may be located at considerable distances from one another so that the best solution is to dispatch a robot team to each designated task area and hope that the team can autonomously complete the task. The robots must then coalesce into teams responsible for each task. The focus of this work is to investigate the various issues that arise while attempting to form multiple-robot coalitions using existing multi-agent coalition formation algorithms. Some solutions are suggested and Shehory and Krauss' (Shehory and Krauss, 1998) multi-agent task allocation scheme algorithm is modified to operate in the multiple-robot domain. This algorithm was chosen because it is designed for DPS Environments, has an excellent real-time response and has been shown to provide results within a bound from optimal.

This paper is organized as follows. Section 2 provides the related work. Section 3 presents an overview of Shehory and Krauss' algorithm. Section 4 identifies issues that entail modification of current coalition formation algorithms. Experimental results are provided in Section 5. Finally, Section 6 discusses the conclusions and future work.

2. Related Work

Shehory and Krauss proposed a variety of algorithms for agent coalition formation that efficiently yield solutions close to optimal. They describe a Kernel oriented model for coalition formation in general environments (Shehory and Krauss, 1996) and non-super additive environments (Shehory and Krauss, 1999). They also provided a computation based algorithm for non-super additive environments (Shehory and Krauss, 1998). Brooks and Durfee (Brooks and Durfee, 2003) provide a novel algorithm in which selfish agents learn to form congregations. Anderson et al. (Anderson et al., 2004) discuss the formation of dynamic coalitions in robotic soccer environments by agents that can learn each other's capabilities. Fass (Fass, 2004) provides results for an Automata-theoretic view of agent coalitions that can adapt to selecting groups of agents. Li and Soh (Li and Soh, 2004) discuss the use of a reinforcement learning approach where agents learn to form better coalitions. Sorbella et al. (Sorbella et al., 2004) describe a mechanism for coalition formation based on a political society.

3. Shehory and Krauss' Algorithm

Shehory and Krauss (Shehory and Krauss, 1998) developed a multi-agent algorithm that is designed for task allocation via agent coalition formation in DPS environments.

3.1 Assumptions

The algorithm includes various assumptions. Assume a set of n agents, $N = A_1, A_2, \dots, A_n$. The agents communicate with each other and are aware of all tasks to be performed. Each agent has a vector of real non-negative capabilities $B_i = \langle b_1^i, b_2^i, \dots, b_r^i \rangle$. Each capability quantifies the ability to perform an action. In order to assess coalitions and task execution, 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 $T = t_1, t_2, \dots, t_m$. A capability vector $B_l = \langle b_1^l, \dots, b_r^l \rangle$ is necessary for the satisfaction of each task t_l . The utility gained from performing the task depends on the capabilities required for its execution. A coalition is a group of agents that decide to cooperate in order to achieve a common task. Each coalition works on a single task. A coalition C has a 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 only if the capability vector necessary for task fulfillment B_t satisfies $\forall 0 \leq i \leq r, b_i^t < b_i^c$.

3.2 The algorithm

The algorithm consists of two primary stages. The first calculates coalitional values to enable comparison of coalitions. The second stage entails an iterative greedy process through which the agents determine the preferred coalitions and form them. Stage one is the more relevant to this work. During this stage the evaluation of coalitions is distributed amongst the agents via extensive message passing, requiring considerable communication between agents. After this stage, each agent has a list of coalitions for which it calculated coalition values. Each agent also has all necessary information regarding the coalition memberships' capabilities. In order to calculate the coalition values, each agent then:

- 1 Determines the eligible coalitions for each task execution t_i by comparing the required capabilities to the coalition capabilities.
- 2 Calculates the best-expected task outcome of each coalition (coalition weight) and chooses the coalition yielding the best outcome.

4. Issues in Multiple-Robot Systems

The algorithm described in Section 3 yields results that are close to optimal. The current algorithm cannot be directly applied to multiple-robot coalition formation. This section identifies issues that must be addressed for multiple-robot domains.

4.1 Computation vs. Communication

Shehory and Krauss's algorithm (Shehory and Krauss, 1998) 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. This work investigates each agent assuming responsibility for all coalitions in which it is a member and thereby eliminating the need for communication. It is necessary to analyze how this would affect each robots computational load. An added assumption is that a robot has a priori knowledge of all robots and their capabilities. Robot capabilities do not typically change; therefore this is not a problem unless a partial or total robot failure is encountered (Ulam and Arkin, 2004). Suppose there are N identical robots and with a perfect computational load distribution, then the number of coalitions each robot must evaluate with communication is:

$$\eta_{with} = \sum_{r=0}^k \binom{n}{r} / n \quad (1)$$

The algorithm distributes coalitions between agents as a ratio of their computational capabilities, adding unwanted complexity. 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 $O(n^{k-1})$ during the calculation-distribution stage. If each agent is responsible for only computation of coalitions in which it is a member, then the number of coalitions evaluated with no communication becomes:

$$\eta_{without} = \sum_{r=0}^{k-1} \binom{n-1}{r} \quad (2)$$

Equation 1 requires fewer computations to evaluate but this is not an order of magnitude difference. In both cases, the agent's computational load is $O(n^k)$ per task. The communicational load per robot is $O(1)$ in the calculation-distribution stage. The additional computation may be compensated for by reduced communication time. The Section 5 experiments demonstrate this point. A desirable side effect is additional fault tolerance. If Robot A fails during coalition list evaluation, values for coalitions containing Robot A are lost and those coalitions are no longer considered. Thus a robot failure does not require information retrieval from that robot. However, the other robots must be aware of the failure so that they can delete all coalitions containing the failed robot.

4.2 Task Format

Current multi-agent coalition formation algorithms assume that the agents have a capability vector, $\langle b_1^i, \dots, b_r^i \rangle$. Multiple-robot capabilities include sensors (camera, laser, sonar, or bumper) and actuators (wheels or gripper). Shehory and Krauss's 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. Robots cannot autonomously exchange capabilities.

Correct resource distribution is also an issue. The box-pushing task can be used to illustrate this point (Gerkey and Mataric, 2002). Three robots cooperate to perform the task, two pushers (one bumper, one camera) and one watcher (one laser, one camera). The total resource requirements are: two bumpers, three cameras, and one laser. However this information is incomplete, as it does not represent the constraints related to sensor locations. Correct task execution requires the laser and camera reside on a single robot. Similarly it is necessary that the

Table 1. Box-pushing task TAM.

	Bumper ₁	Bumper ₂	Camera ₁	Camera ₂	Camera ₃	Laser ₁
Bumper ₁	X	0	1	0	0	0
Bumper ₂	0	X	0	1	0	0
Camera ₁	1	0	X	0	0	0
Camera ₂	0	1	0	X	0	0
Camera ₃	0	0	0	0	X	1
Laser ₁	0	0	0	0	1	X

bumper and laser reside on different robots. This implies that simply possessing the adequate resources does not necessarily create a multiple-robot coalition that can perform a task, other locational constraints have to be represented and met.

A matrix-based constraint representation is proposed for the multiple-robot domain in order to resolve the problem. The task is represented via a capability matrix called a Task Allocation Matrix (TAM). Each matrix entry corresponds to a capability pair (for example [sonar, laser]). A 1 in an entry indicates that the capability pair must reside on the same robot while a 0 indicates that the pair must reside on separate robots. Finally an X indicates a do not care condition and the pair may or may not reside on the same robot. Every coalition must be consistent with the TAM if it is to be evaluated as a candidate coalition. The box-pushing TAM is provided in Table 1. The entry (Laser₁, Camera₃) is marked 1, indicating that a laser and a camera must reside on the same robot. Similarly the (Bumper₁, Laser₁) entry is marked 0 indicating the two sensors must reside on different robots.

The TAM 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 machine or different machines. A constraint graph can be drawn with locational constraints represented as arcs labeled *s* (same robot) or *d* (different robot). Another constraint is the resource constraint representing that a robot only have as many instances of a sensor and actuator as indicated by the associated capability vector. A robot with one camera can only be assigned one camera node in the constraint graph. Thus all sensors and actuators of the same type have a resource constraint arc labelled *r* between them.

Figure 1 provides the box-pushing task constraint graph. This task's resource constraints between Bumper₁ and Bumper₂ are implied by their

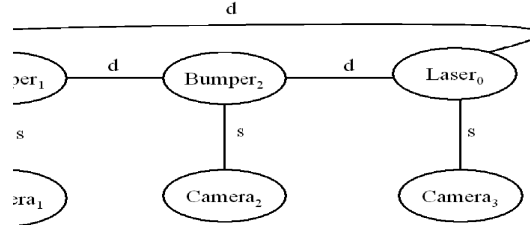


Figure 1. Box-pushing task constraint graph

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 $Bumper_2$. Similarly the resource constraints between $Camera_1$, $Camera_2$ and $Camera_3$ are implied by the locational constraints between them and there is no need to test them separately. Hence the absence of edges labeled r .

The domain values for each variable in the CSP formulation in Figure 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 is left with an empty domain value set then the current assignment has failed and the current coalition is deemed infeasible. A successful assignment indicates the sub-task to which each robot was assigned.

Using the 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. This is verified by the experimental results in Section 5.

4.3 Coalition Imbalance

Coalition imbalance or lopsidedness is defined as the degree of unevenness of resource contributions made by individual members to the coalition, a characteristic 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 (coalition weight) as a coalition with evenly distributed capabilities, since 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. Over reliance on dominating members can

cause task execution to fail or considerably degrade. If a robot is not a dominating member then it is more likely that another robot with similar capabilities can replace it.

Rejecting lopsided coalitions in favor of balanced ones is not straightforward. When comparing coalitions of different sizes, there can arise a subtle trade-off between lopsidedness and the coalition size. The argument may be made both for fault tolerance and for smaller coalition size. It may be desirable to have coalitions with as few robots as possible. Conversely, there may be a large number of robots thus placing the priority on fault tolerance and balanced coalitions. The Balance Coefficient metric is introduced to quantify the coalition imbalance level. In general, if a coalition has a resource distribution (r_1, r_2, \dots, r_n) , then the balance coefficient for that coalition with respect to a particular task can be calculated as follows

$$BC = \frac{r_1 \times r_2 \times \dots \times r_n}{\left[\frac{taskvalue}{n}\right]^n} \quad (3)$$

A perfectly balanced coalition has a coefficient of 1. The question is how to incorporate the balance coefficient into the algorithm in order to select better coalitions. As previously discussed two cases arise:

- 1 *Sufficient number of robots and high fault tolerance:* Initially the algorithm proceeds as in Section 3, 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 balance coefficient for all these coalitions and chooses the most balanced coalition. This ensures that the algorithm always favors the balanced coalition.
- 2 *Economize on the number of robots:* Maintain a list of all coalitions with values within a bound of the best coalition value. Remove all coalitions larger than the best coalition from the list. Select the coalition with the highest balance coefficient.

5. Experiments

Three experiments testing the validity of the algorithm modifications were conducted, each highlighting a suggested modification. The first experiment measured the variation of time required to evaluate coalitions with and without communication. The number of agents and maximum coalition size were both fixed at five. Communication occurred via TCP/IP sockets over a wireless LAN (see Figure 2). The time for coalition evaluation without communication is significantly less than the time

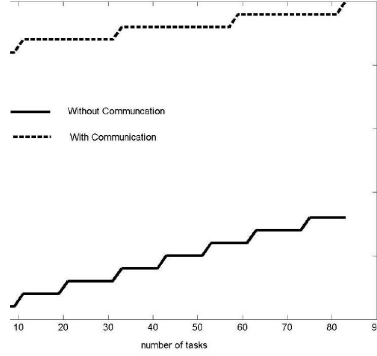


Figure 2. Execution time with and without communication

required for evaluation with communication. The time without communication increases at a faster rate as the number of tasks increases. This result occurs because the agent must evaluate a larger number of coalitions when it forgoes communication. Presumably, the two conditions will eventually meet and thereafter the time required with communication will be less than that required without communication. For any practical Agent/Task ratio the time saved by minimizing communication outweighs the extra computation incurred.

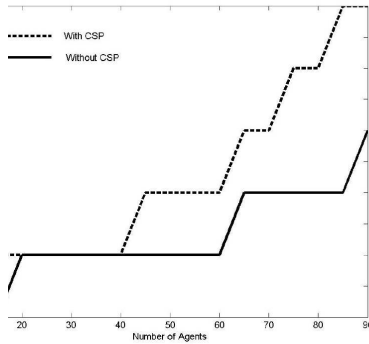


Figure 3. Execution time vs. Number of Agents

The second set of experiments measured the effect of the CSP formulation on the algorithm's execution time. This experiment demonstrates the algorithm's scalability. Figure 3 measures the variation of execution time against the number of agents both with and without constraint checking in the constraint satisfaction graph. Figure 4 shows the variation of execution time compared to the number of tasks. The task

complexity in these experiments was similar to the box-pushing task. It can be seen from Figures 3 and 4 that the CSP formulation does not add a great deal to the algorithm’s execution or running time. This implies that this formulation can be used to test the validity of a multiple-robot coalition without incurring much overhead.

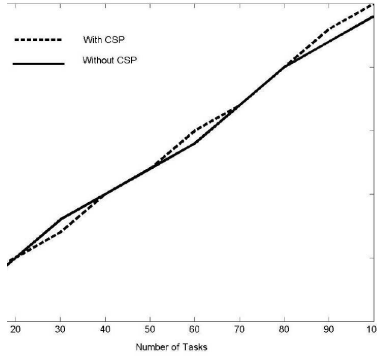


Figure 4. Execution time vs. Number of Tasks

The third set of experiments demonstrates the effect of utilizing the Balance Coefficient to favor the creation of balanced coalitions. The Player/Stage simulation environment (Gerkey et al., 2003) was employed for this experiment. The simple tasks required breaking up a formation of resized hockey pucks by bumping into the formation. The degree of task difficulty was adjusted by varying the hockey pucks’ coefficient of friction with the floor. Adjusting the forces they could exert varied the robots’ capabilities. There are no locational constraints on the task capability requirements. Ten simulated robots were used for the experiment, as shown in Figure 5. The robots were numbered 1 to 10 from the top of the figure along left side. Each robot had a specific capability type: small robots had 10 units of force (robots 1, 2, 8, 9, 10), medium sized robots had 15 units of force (robots 5, 6, 7) and large robots had 20 units of force (robots 3, 4). Simulation snapshots are provided for a task requiring 50 units of force. Figure 5 shows the formed coalition without balancing. The coalition is comprised of two large robots and one small robot.

Figure 6 shows the same task incorporating the balance coefficient into the coalition formation. This choice places a low priority on fault tolerance and a high priority on economizing the number of robots (Case 1 from Section 4.3). The formed coalition is comprised of two medium sized robots and one large robot. The resulting coalition is more balanced and has a higher balance coefficient (0.972 as opposed to 0.864 for



Figure 5. Two large robots and one small robot form a coalition

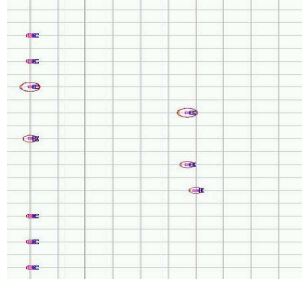


Figure 6. One large and two medium-sized robots form a coalition

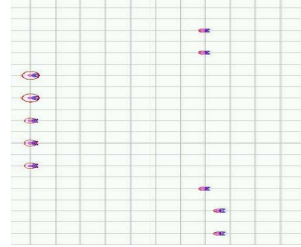


Figure 7. Five small robots form a coalition

the coalition in Figure 6). Figure 7 depicts the experiment conducted with no restrictions on the coalition size (Case 2 from Section 4.3). The resulting coalition consists of five small robots. Thus a perfectly balanced coalition (balance coefficient = 1) is obtained when the coalition size is unconstrained. 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 find a replacement as opposed to replacing a larger (more capable) robot.

6. Conclusion and Future Work

Finding the optimal multiple-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 multiple-robot domain. This paper identifies modifications and incorporates them into an existing multi-agent coalition formation algorithm. The impact of extensive communication between robots was shown to be severe enough to endorse relinquishing communication in favor of additional computation when possible. The task format in multi-robot coalitions was modified to adequately represent additional con-

straints imposed by the multiple-robot domain. The concept of coalition imbalance was introduced and its impact on the coalition's fault tolerance was demonstrated.

Further algorithm modifications will permit more complex task execution by utilizing a MURDOCH (Gerkey and Mataric, 2002) style task allocation scheme within coalitions. A future goal is to investigate methods of forming coalitions within a dynamic real-time environment. The long-term goal is to develop a highly adaptive, fault tolerant system that would be able to flexibly handle different tasks and task environments.

References

- Anderson, J. E., Tanner, B., and Baltes, J. (2004). Dynamic coalition formation in robotic soccer. Technical Report WS-04-06, AAAI workshop.
- Brooks, C. H. and Durfee, E. H. (2003). Congregation formation in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 79:145–170.
- Fass, L. (2004). An automatic-theoretic view of agent coalitions. Technical Report WS-04-06, AAAI workshop.
- Gerkey, B. and Mataric, M. (2002). Sold! auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18:758–68.
- Gerkey, B. and Mataric, M. (2004). A framework for studying multi-robot task allocation. *International Journal of Robotics Research*. to appear.
- Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *11th Intr. Conf. on Advanced Robotics*, pages 317–323.
- Li, X. and Soh, L.-K. (2004). Investigating reinforcement learning in multiagent coalition formation. Technical Report WS-04-06, AAAI workshop.
- Parker, L. (1998). Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14:220–240.
- Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tomhe, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111:209–238.
- Shehory, O. and Krauss, S. (1996). A kernel oriented model for coalition-formation in general environments: Implementation and results. In *AAAI*, pages 134–140.
- Shehory, O. and Krauss, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence Journal*, 101:165–200.
- Shehory, O. and Krauss, S. (1999). Feasible formation of coalitions among autonomous agents in non-super-additive environments. *Computational Intelligence*, 15:218–251.
- Sorbella, R., Chella, A., and Arkin, R. (2004). Metaphor of politics: A mechanism of coalition formation. Technical Report WS-04-06, AAAI workshop.
- Ulam, P. and Arkin, R. (2004). When good comms go bad: Communications recovery for multi-robot teams. In *2004 IEEE Intr. Conf. on Robotics and Automation*, pages 3727–3734.
- Werger, B. and Mataric, M. (2000). Broadcast of local eligibility: Behavior-based control for strongly-cooperative robot teams. In *Autonomous Agents*, pages 347–356.