Self-Reconfiguration in Modular Robots Using Coalition Games with Uncertainty

Zachary Ramaekers[1], **Prithviraj Dasgupta**[1],
Vladimir Ufimtsev,[1] **S. G. M. Hossain,**[2] **Carl Nelson**[2]
[1]Computer Science Department, University of Nebraska, Omaha
[2]Mechanical Engineering Department, University of Nebraska, Lincoln

Abstract

We consider the problem of dynamic self-reconfiguration in a modular self-reconfigurable robot (MSR). Previous MSR self-reconfiguration approaches search for new configurations only within the modules of the MSR that needs reconfiguration. In contrast, we describe a technique where an MSR that needs to reconfigure communicates with other MSRs in its vicinity to determine if modules can be shared from other MSRs, and then determines the best possible configuration among the combined set of modules. We model the MSR self-reconfiguration problem as a coalition structure generation problem within a coalition game theoretic framework. We formulate the coalition structure generation problem as a planning problem in the presence of uncertainty and propose an MDP-based algorithm to solve it. We have implemented our algorithm within an MSR called ModRED that is simulated on the Webots simulation platform. Our results show that using our self-reconfiguration algorithm, when an MSR needs to reconfigure, a new configuration that is within $5 - 7\%$ of the globally optimal configuration can be determined. We have also shown that our algorithm performs comparably with another existing algorithm for determining optimal coalition structure.[1]

Introduction

Over the past few years, modular self-reconfigurable robots (MSRs) have emerged as an important research direction in robotic systems (Yim and et al. 2007). MSRs are made up of functionally simple modules that are capable of working together. On its own, each module is capable of performing very limited operations, but when connected with other modules, they can adapt their shape to accomplish complex tasks. In spite of their simple and inexpensive construction, and easy maneuverability, a principal challenge in MSRs is to solve the self-reconfiguration problem — how to adapt their shape autonomously so that they can change tasks or continue their operation after encountering obstacles or occlusions that impede their movement. This problem is challenging because a fixed set of rules does not work for all situations. An MSR needs to perceive its current environment to determine how many modules to connect together, and the configuration or shape those modules should get into, so that the MSR can perform its assigned task most efficiently. In this paper, we have addressed the MSR self-reconfiguration problem by modeling it as a coalition structure generation problem in coalition game theory. Coalition games are suitable for the MSR self-reconfiguration problem because the solution found by a coalition game ensures stability — once the best partition or coalition of agents, corresponding to the best configuration of MSRs has been found, the MSR modules that have been determined to form the new configuration will remain together and will not try to leave the new configuration and attempt to combine with other modules. However, there are several research challenges that need to be addressed while using coalition game theory for MSR self-reconfiguration. First, in coalition game theory, the assimilation of agents into teams and the communication between agents is assumed to be free of cost. However, for MSRs, modules incur "cost" by expending energy to communicate with each other and physically move to each other's proximity to dock with each other. Secondly, in coalition games, the values or utilities the agents calculate for determining how much they benefit by participating in a coalition is assumed to be free from uncertainty. In contrast, in MSRs, due to the presence of noise in the robot's sensor readings and due to limited sensor ranges, the value of a coalition is not 100% certain. To address these problems, in this paper we describe a novel technique that combines coalition game theory with planning under uncertainty using Markov Decision Processes (MDPs). To illustrate the operation of our MSR we have used the domain of robotic exploration of initially unknown environments. Our experimental results show that coalition game theory-based algorithms can be successfully used to dynamically self-reconfigure an MSR called ModRED into different configurations. We have compared our results while using three different pruning heuristics for our algorithm and shown that MSR modules using our techniques to self-reconfigure receive $2 - 10\%$ more reward as compared to MSR modules using a previously existing algorithm for reconfiguring. To the best of our knowledge, this is one of the first works that combines coalition game theory and planning under uncertainty to address

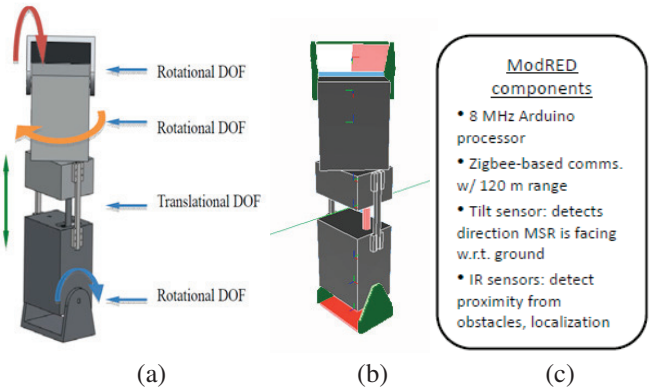
[1]This research has been supported by a NASA Nebraska research mini-grant.

the problem of self-reconfiguration in MSRs.

Related Work

Modular self-reconfigurable robots (MSRs) are a type of self-reconfigurable robots that are composed of identical modules. These modules can change their connections with each other to manifest different shapes of the MSR and select a shape that enables the MSR to perform its assigned task efficiently (Castano, Shen, and Will 2000; Stoy, Brandt, and Christensen 2010). An excellent overview of the state-of-the-art MSRs and related techniques is given in (Yim and et al. 2007). Out of the three types of MSRs — chain, lattice and hybrid - we have used a chain-type MSR to illustrate the experiments in this paper although our techniques could be used for other types too. The self-reconfiguration problem in MSRs has been solved using search-based (Butler, Brynes, and Rus 2001; Chirikjian, Pamecha, and Ebert-Upfhoff 1996) and control-based techniques (Rosa et al. 2006). However, both these techniques require the initial and goal configuration to be determined before the reconfiguration process starts. A third technique called task-based reconfiguration has recently shown considerable success (Kamimura et al. 2008). Here the goal configuration of an MSR doing reconfiguration is not determined *a priori*, but is determined as the configuration that helps the MSR perform its task efficiently. We have used such a task-based reconfiguration technique where the configuration that an MSR will get into after reconfiguring is determined as the configuration that gives the MSR its highest expected efficiency in performing its intended task. The specific reconfiguration procedure used by the MSR uses a multi-agent, coalition game theory based technique called coalition structure generation.

Coalition game theory gives a set of techniques that can be used by a group of agents to form teams or coalitions with each other (Myerson 1997). A coalition can be loosely defined as a set of agents that remain together with the intention of cooperating with each other, possibly to perform a task. In terms of MSRs a coalition represents a set of MSR-modules that are connected together while performing a certain task. Within coalition games, the coalition structure generation problem that deals with partitioning the agents into disjoint and exhaustive sets called coalitions has received significant attention. This problem is NP-complete, and Sandholm (Sandholm et al. 1999) and Rahwan (Rahwan 2007) have proposed anytime algorithms to find near-optimal solutions. However, none of these techniques incorporate uncertainty in the values of nodes of the coalition structures. In contrast, while modeling MSR reconfiguration as a coalition structure generation problem, it makes sense to incorporate uncertainty in the value of the combinations of MSR modules due to sensor noise and limited sensor ranges. To do model the uncertainty in the coalition structure generation problem, we propose using a Markov Decision Process (MDP)- a mathematical model for representing uncertainty in planning problems by assigning probabilities to transitions between states (Puterman 1995). Complimentary to our research, Chalkiadakis (Chalkiadakis 2007) has proposed a technique to model uncertainty in coalition games.



(a) (b) (c)

Figure 1: (a) CAD figure of a single module of the ModRED robot. (b) A simulated version of ModRED inside Webots. (c) Major components of the ModRED robot.

ModRED: A Novel 4-DOF Modular Robot

We have used an MSR called ModRED (**Mod**ular **R**obot for **E**xploration and **D**iscovery) (Chu, Hossain, and Nelson 2011) that is currently being developed by us, for implementing and testing the techniques in this paper. Unlike most other MSRs, ModRED has a novel 4-th rotational DOF that allows each module to rotate along its long axis, as shown in Figure 1(a). The additional rotational DOF enables ModRED to rotate sideways and possibly maneuver itself out of tight spaces. A picture of the ModRED robot, its simulated version within Webots and its major components are shown in Figure 1. For the simulated version of each module, we have used a GPS node that gives global coordinates on each robot[2], an accelerometer to determine the alignment of the robot with the ground, in addition to the IR sensors and Zigbee modules in the physical robot. The movement of ModRED in fixed configuration is enabled through gait tables (Stoy, Brandt, and Christensen 2010). Each gait table applies to a specific movement of the robot in a specific configuration. The contents of the gait table give the sequence of movements of the different joints of the robot to achieve the desired motion. Videos showing the movement of the ModRED robot in different configurations using gait tables are available at http://cmantic.unomaha.edu/modred. While moving in a fixed configuration, if ModRED's motion gets impeded by an obstacle or an occlusion in its path, it needs to reconfigure into a new configuration so that it can continue its movement efficiently. In the next section, we formalize the MSR self-reconfiguration problem and then provide a coalition game-based algorithm to solve it in the presence of uncertainty.

Dynamic Self-Reconfiguration in MSRs

Let A be the set of modules or agents that have been deployed in the environment. The set of MSRs at time t, $\{A_i^t\}$, is defined as a set of exhaustive and disjoint partitions of A,

[2]In the physical ModRED robot, relative positioning is planned to be done using the IR sensors.

i.e., $\cup_i A_i^t = A$. The i^{th} MSR at time t is given by a set of ordered modules or agents, i.e.,

$$A_i^t = \{a_{i,1}^t, a_{i,2}^t, a_{i,3}^t, ..., a_{i,|A_i^t|}\} \quad (1)$$

Using this definition, when A_i^t is a singleton, it represents a single module that is not coupled with any other modules.

Let C denote a finite set of configuration types for an MSR (e.g, chain, ring, tree, etc.), $S \in \mathbb{Z}$ denote the set of sizes of the MSR . Taken together $C \times S$ denotes the space of configurations that an MSR can take. For an MSR A_i[3], with size $\mid A_i \mid$, the configuration function, $conf(A_i) = (c, s)$ where $c \in C$ and $1 < s <= \mid A_i \mid$ gives the set of possible configurations. Let Θ denote a finite set of task types that an MSR can perform. The efficiency with which a certain MSR performs its task is characterized by its configuration, its size and the type of task it is trying to perform. The efficiency is captured by an efficiency function, $E : C \times S \times \Theta \rightarrow \mathbb{R}$. The exact value of the efficiency function is dependent on the task type. For example, for a robotic exploration task this efficiency can be determined from attributes that consider the area of the region covered by the MSR, the time required to cover the region, the number of movements of the MSR's actuators and the energy spent in moving the MSR, the number of failed movements of the actuators, etc. While performing task $\theta \in \Theta$ in configuration (c, s), an MSR records the efficiency of performing the task at intervals of T timesteps and maintains an average value of the efficiency $\bar{E}(c, s, \theta)$, within a hash map $H_E : C \times S \times \theta \rightarrow \bar{E}$. These average efficiency values from the hash map are used later on in the self-reconfiguration calculations.

An MSR moving in a configuration (c_{old}, s_{old}) needs to reconfigure when the avearage efficiency for doing its task θ falls below a certain thresold $E_{threshold}$. This situation can happen, for example, when the MSR gets stuck at an obstacle while navigating during an exploration task. The MSR then needs to determine a new configuration (c_{new}, s_{new}) to see if the average efficiency of performing its intended task θ in the new configuration improves to a value above the threshold $E_{threshold}$. To do this the MSR considers the average efficiency that it is likely to receive in the new configuration from its hash map, $H_E(c_{new}, s_{new}, \theta)$, as well as the cost required to do the reconfiguration. We parameterize the reconfiguration cost in the following manner. Let $cost(A_\alpha, A_\beta)$ denote the cost that will be incurred to couple MSRs A_α and A_β with each other. If these two MSRs are already part of the same MSR, this cost is 0. Otherwise, this cost is given by the sum of the costs of undocking the appropriate modules in A_α and A_β respectively from their current MSRs, the cost of one of the MSRs, say A_β, moving to the vicinity of the other module A_α and the cost of aligning and docking the two MSRs, as described below:

[3]For legibility, and without loss of generality, we drop the notation for time t from the MSR

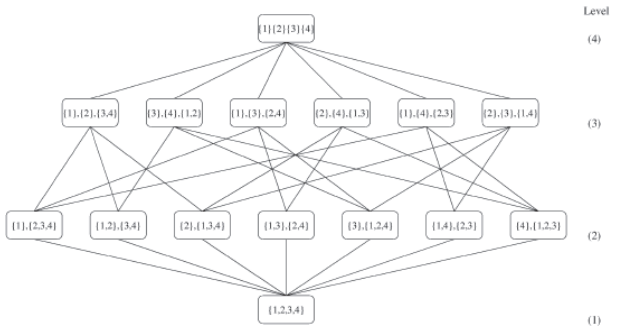


Figure 2: Coalition Structure graph with 4 agents, when there is no uncertainty in the agents' actions.

$$cost(A_\alpha, A_\beta) = \begin{cases} 0, & \text{if } A_\alpha, A_\beta \in A_k \\ cost_{Undock}(A_\alpha) & \\ + cost_{Undock}(A_\beta) & \\ + cost_{Crawl}(A_\beta, loc(A_\alpha)) & \\ + cost_{AlignAndDock}(A_\alpha, A_\beta), & \text{otherwise} \end{cases} \quad (2)$$

Let A_i denote an MSR that needs to reconfigure. Before starting the reconfiguration, A_i determines via request-response protocol if there are othera MSRs that are within its communication range and could participate in A_i's reconfiguration by potentially providing some of their modules to A_i to form its new configuration. We call the set of MSRs pariticipating in the reconfiguration of MSR A_i, including A_i itself, as A_i's *reconfiguration participation set* (RPS), i.e., if there are $n-1$ other MSRs responding to A_i's request to reconfigure, then $RPS(A_i) = \{A_1, A_2, ...A_{i-1}, A_i, A_{i+1}, ...A_n\}$, (the numbering of the MSRs is arbitrary).

Within this framework, we define the MSR self-reconfiguration problem as the following: Let A_i be an MSR that needs to reconfigure and $RPS(A_i)$ be its corresponding RPS. Recall that each $A_j \in RPS$ is an ordered set of modules, $A_j = \{a_{j,1}, a_{j,2}....\}$ as given by Equation 1.

Definition. **Modular Self Reconfiguration.** Find a new partition of the modules among the members of $RPS(A_i)$, given by $A'_1, A'_2.....A'_n$ such that

$$\sum_{i=1}^{n'} \left[\bar{E}(conf(A'_i), \theta) - \sum_{A_\alpha, A_\beta \in A'_i} cost(A_\alpha, A_\beta) \right] \quad (3)$$

is maximized. This condition ensures that the new partition among the modules in $RPS(A_i)$ get the highest expected efficiency while incurring the least cost to reconfigure.

Coalition Game for MSR Self-Reconfiguration

We propose a novel coalition game based approach to address the MSR self-reconfiguration problem. Each module of the MSR is provided with a software agent that performs calculations related to the coalition game based algorithm to solve the modular self reconfiguration problem. We have used a popular representation of coalition games called

characteristic function games (CFG) (Myerson 1997). A CFG is defined by a pair of attributes (N, v), where N is the set of agents, and $v : 2^N \rightarrow \mathbb{R}$ is called a characteristic function or value function. v gives a real number called the value or worth for each possible subset or coalition S of the set of N agents. Recall that for N agents, the number of possible subsets or coalitions $S \subseteq N$ is given by size of the power set of N minus the null set, i.e., $\mid S \mid = 2^N - 1$. A *coalition structure* is an enumeration of the partitions S of N such that every agent appears exactly once in one of the partitions. For a set of N agents, let CS_N denote the set of coalition structures. For example, with $N = 3$, $CS_3 = (\{1\}\{2\}\{3\}, \{1\}\{2,3\}, \{2\}\{1,3\}, \{3\}\{1,2\}, \{1,2,3\})$. CS_N can be enumerated recursively as a coalition structure graph (CSG), as shown in Figure 2. Each coalition structure $CS_{N,i} \in CS_N$ appears as a node in the CSG. Nodes are organized into levels, and a node at level $l-1$ can be generated recursively by combining pairwise the members from disjoint partitions, for each node in level l. Each coalition structure $CS_{N,i}$ is associated with a value $V(CS_{N,i})$ that is usually calculated by adding the values of each coalition within the coalition structure, i.e., $V(CS_{N,i}) = \sum_{S \in CS_{N,i}, S \subseteq N} v(S)$. For example, with 4 agents, for the coalition structure $\{1,2\}\{3\}\{4\}$, $V(\{1,2\}\{3\}\{4\}) = v(\{1,2\}) + v(3) + v(4)$. For the context of MSR reconfiguration, an agent i corresponds to a single MSR-module a_i, a coalition S corresponds to an MSR A_i, while a coalition structure corresponds to a set of MSRs. The value of a coalition $v(S)$ is given by $v(S) = \bar{E}(conf(A'_i), \theta) - \sum_{A_\alpha, A_\beta \in A'_i} cost(A_\alpha, A_\beta)$, as given in Equation 3. Using this correspondence, to solve the MSR self-reconfiguration problem given in Definition 1, we have to find the coalition structure in the CSG that corresponds to the maximum value, i.e., find $CS^* = arg\max_{CS_{N,i}} V(CS_{N,i})$.

MDP-based uncertainty model for MSR Reconfiguration Cost

The components of the reconfiguration cost given in Equation 2, comprising the undocking and docking costs of modules, are dependent on the construction of the modules and can be estimated as a constant with a certain variance. In contrast, $cost_{Crawl}$ in Equation 2 corresponds to the cost incurred by a module to move from an initial to a goal location. This cost is uncertain and cannot be calculated *a priori*. It depends on the environment features (e.g., ridges or chasms that are difficult to maneuver) in the region between the two modules. These features might not be possible to detect by the modules at the time of their decision-making to move close to each other to connect, due to their limited sensor range. Additionally, localization noise in determining the position of the MSR modules planning to connect together can contribute to uncertainty in calculating the value of $cost_{Crawl}$. This situation corresponds to a path planning problem in the presence of uncertainty, for the two modules trying to move to each other's vicinity. In the coalition game model of the reconfiguration problem, each possible partition of $RPS(A_i)$ corresponds to a coalition structure, i.e., a node in the CSG. And the uncertainty in the cost incurred by two MSRs in connecting with each other implies that there is uncertainty in the value of each node in the CSG. Following uncertainty models in game theory (Myerson 1997), we have modeled this uncertainty in a CSG node's value by allowing each node to have different types. Each node's type is drawn from a finite set of types $\Phi = \{$sub-additive, additive, super-additive$\}$ and the distribution of types is given by $f(\phi)$. The value function with types V_ϕ is defined as below:

$$V_\phi(CS_{N,i}, \phi) = \begin{cases} V(CS_{N,i})(1 - f(\phi)), & \text{if } \phi\text{=sub-additive} \\ V(CS_{N,i}), & \text{if } \phi\text{=additive} \\ V(CS_{N,i})(1 + f(\phi)), & \text{if } \phi\text{=super-additive} \end{cases} \quad (4)$$

Generating and Pruning the Coalition Structure Graph. To generate the CSG we start with the node at level n (top of the graph), where each agent is a singleton. At each node, we generate three values corresponding to the sub-additive, additive and super-additive types. From each node at level l, children nodes at level $l-1$ are generated recursively by pairwise merging of the sets. To keep the number of nodes in the CSG from becoming exponential, we have used three strategies to prune the CSG nodes while generating the nodes from level l in level $l-1$: *Strategy 1.:* At level $l-1$, keep the child with the highest additive node value and the two children with the two lowest additive node values. *Strategy 2:* At level $l-1$, keep three children with the highest, median and lowest additive node values, *Strategy 3:* At level $l-1$, keep three randomly chosen children nodes. In the first two strategies, we retain the child with the highest node value because it will be selected at level $l-1$ while searching for the optimal coalition structure node. However, the highest valued node at level $l-1$ might not lead to the optimal valued node for the entire CSG. In fact, lower valued nodes from level $l-1$ might lead to the optimal node that is at a lower level. To account for this, we select two additional nodes in each of our pruning strategies.

Determining Optimal Policy in Pruned CSG. Let $\widehat{CS}_N$ denote the set of nodes in a CSG CS_N after it has been pruned. After the CSG has been pruned, our objective is to traverse this pruned CSG and find the node $CS^* \in \widehat{CS}_N$ that has the optimal value. However, finding the optimal valued node is not straightforward because each node can take three values corresponding to its three types. The realization of the type of a node happens at runtime, and, therefore, there is no guarantee that the optimal node will be reached if a fixed path towards it, from an initial state is followed. Instead, in our scenario, we have to find a policy that gives a suggested next node to visit for each node in the pruned CSG. We do this by modeling the pruned CSG as an MDP $=< St, Ac, Tr, R >$. The nodes of the pruned CSG correspond to the state space of the MDP, i.e., $St = \widehat{CS}_N$, while an action in the MDP, $ac \in Ac$ transitions the search from from one node in the CSG to another, i.e., $s \xrightarrow{ac} s'$ where $s, s' \in St$. The transition probability Tr is given by $Tr(s', ac, s) = Pr(s' \mid s, ac) = f(\phi)$, where the type of s' is ϕ and $s, s' \in St, ac \in Ac$. Finally, the reward in a state $s \in St$ is given by $R(s) = V_\phi(CS_{N,i}, \phi)$, where state s

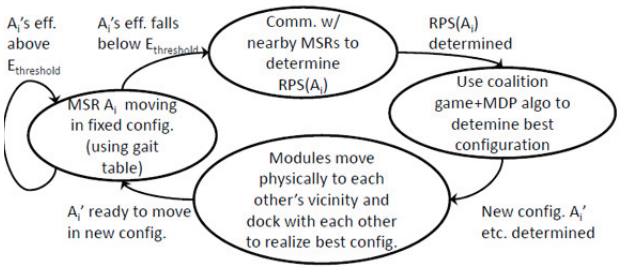


Figure 3: Controller for the ModRED for maneuvering in fixed configuration and self-reconfiguring using coalition game theory and MDP-based algorithm.

in the MDP corresponds to the CSG node $CS_{N,i}$. The optimal policy π^* is determined by solving the Bellman equation corresponding to value iteration in the MDP:

$$U'(CS_{N,i}) = R(CS_{N,i}) + \gamma(max_{j \in m} U(CS_{N,j}))$$

where γ is a discount factor, and m is the set of all neighbor nodes of node i in the CSG. When we conclude the value iteration, each node has a corresponding utility value U' with it.

The final step in finding the optimal node in the CSG is to traverse the CSG using the policy π^* determined by the MDP's value iteration. We begin the CSG traversal by starting at a CSG node that corresponds to the current configurations of the MSRs given in $RPS(A_i)$. For example, if $RPS(A_i) = \{A_1, A_2\}$, where A_1 needs to reconfigure and $A_1 = \{a_{1,1}, a_{1,2}\}$ and $A_2 = \{a_{2,1}, a_{2,2}\}$, then we start with the CSG node $(\{1,2\}\{3,4\})$ corresponding to the current configuration of the MSRs. At each node, we select the next node to visit as given by the policy. The node's value V_ϕ is recorded if it is the highest value, max, seen among the nodes visited thus far. We continue traversing the graph until either all nodes of the graph have been traversed or we haven't seen a node that has a value greater than max among the last k nodes. The coalition structure node that has the maximum value max when the traversal terminates, is returned as the optimal coalition structure node. The solution to the self-reconfiguration problem is given by the MSR configurations corresponding to this node's coalition structure.

Figure 3 shows an abstract view of the controller used to integrate the coalition game-based self-reconfiguration technique with the conventional gait table-based maneuver of the MSR. When an MSR A_i has to perform tasks like navigation, it uses the gait table corresponding to its configuration to maneuver itself. However, if this operation fails because of getting impeded by an obstacle or occlusion, A_i first determines the reconfiguration participation set $RPS(A_i)$ by communicating with nearby MSRs. It then runs the coalition game and MDP-based algorithm to determine the best configuration. Finally, the modules required to connect with each other to realize the best configuration move to each other's vicinity and dock with each other.

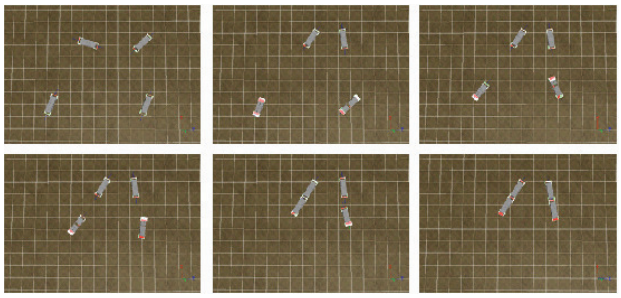


Figure 4: Sequence of movements of four single module MSRs determining the best configuration and getting into two two-module chain configurations.Video available at http://cmantic.unomaha.edu/modred

Experimental Results

We have tested our coalition game based self-reconfiguration algorithm on the simulated version of ModRED within Webots. Screen captures from a self-reconfiguration scenario done by 4 modules is shown in Figure 4. We begin by placing a group of 4 single module ModRED MSRs oriented in different directions within a simulated environment. One of the modules is identified randomly as an in-charge module that runs our algorithm for optimal coalition structure generation under uncertainty. When the algorithm is complete, the module in charge has a set of all the coalitions that should be formed. For each resulting MSR a leader module is selected by the in-charge module to coordinate the docking between the other modules in its MSR. As can be seen from the images, in this instance the best coalition structure consisted of two chains of modules each containing two modules. The modules oriented themselves properly and were able to form two chains.

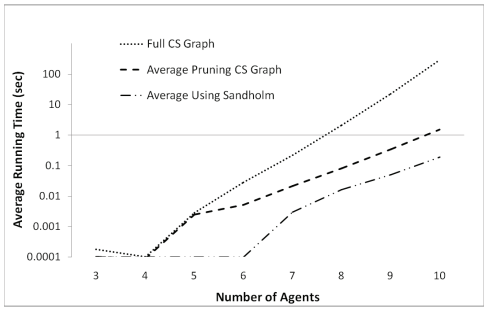


Figure 5: Average time to compute the CSG

For comparison of our approach, we have implemented Sandholm's optimal coalition structure generation algorithm (Sandholm et al. 1999). In this algorithm, we search only the two bottommost levels of a CSG and return the node with the highest value found within those two layers as the coalition structure corresponding to the solution of the MSR self-reconfiguration problem. All results were averaged over 10 simulation runs.

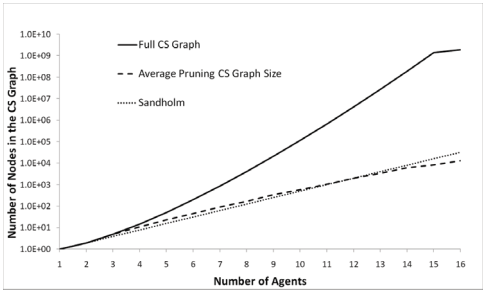


Figure 6: Average number of nodes in the CSG

For our first set of experiments, we determined the time required to generate the CSG. Figure 5 shows the average time required to generate the CSG for different numbers of agents while using no pruning, median pruning (pruning strategy 2 above) and Sandholm's technique. The average number of nodes generated for the three different techniques are shown in Figure 6. We observe that Sandholm's technique takes the least time to construct the CSG and also generates the least number of nodes because it confines its search only to the two bottommost levels of the CSG. However, the difference in the number of nodes generated between our pruning strategy and Sandholm's technique is nominal.

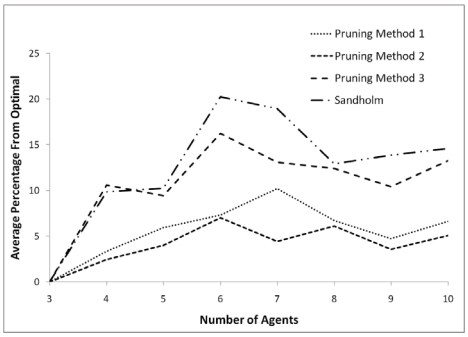


Figure 7: Average percent difference from optimal value

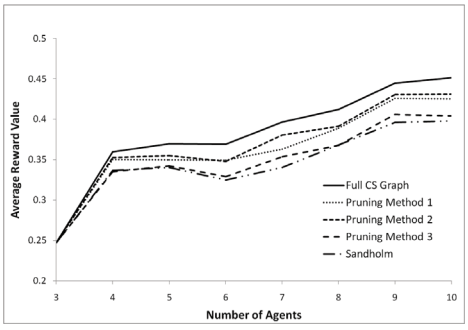


Figure 8: Average optimal reward values

For our next set of experiments, we quantified how well the coalition structure determined by our technique compares with the optimal coalition structure in the full CSG. Figure 7 shows the results of this experiment for $2 - 10$ agents. We observe that using our pruning strategies 1 and 2, the difference between the values of the coalition structure determined by our algorithm and the optimal coalition structure is $5-7\%$. In contrast, the compared Sandholm's algorithm and pruning strategy 3 give a difference of $10-20\%$ from the optimal value. The reason for the better performance of our first two pruning strategies is that those strategies retain nodes in every level of the CSG, as compared to Sandholm's strategy that retains only the lowermost two levels of the CSG. Our third pruning strategy that retains randomly selected nodes at each level also performs poorly because the nodes with the best values that are closer to the global optimal value might be omitted due to the random selection strategy. For our final set of experiments, we show the average optimal reward values received by each strategy. As expected, the reward values with the full CSG are highest. However, our first two pruning strategies perform comparably with only about 5% decrease in the reward value, owing to their ability of retaining some nodes at every level of the CSG. The third pruning strategy and Sandholm's technique get rewards that are about 7.5% below the first two pruning strategies because of their random selection of nodes and retaining limited number of nodes in the CSG.

Conclusion and Future Work

In this paper, we developed a coalition game theory based algorithm for MSR self-reconfiguration and validated it to work on a simulated MSR called ModRED. There are several future directions we are investigating including the investigation and refinement of distributed models of planning under uncertainty, simulating simulate exploration and coverage-like tasks using our algorithms on realistic terrains. and implementing the algorithms proposed in this paper on the hardware of the ModRED robot.

References

Butler, Z.; Brynes, S.; and Rus, D. 2001. Distributed motion planning for modular robots with unit decompressable modules. In *IEEE/RSJ Intl. Conf. Intell. Rob. and Sys.*, 790–796.

Castano, A.; Shen, W.; and Will, P. 2000. Conro: Towards deployable robots with inter-robots metamorphic capabilities. *Autonomous Robots* 8:309–324.

Chalkiadakis, G. 2007. *A Bayesian Approach to Multiagent Reinforcement Learning and Coalition Formation under Uncertainty*. Ph.D. Dissertation, University of Toronto.

Chirikjian, G.; Pamecha, A.; and Ebert-Upfhoff, I. 1996. Evaluating efficiency of self reconfiguration in a class of modular robots. *Robotics Systems* 13:317–338.

Chu, K.; Hossain, S. G. M.; and Nelson, C. 2011. Design of a four-dof modular self-reconfigurable robot with novel gaits. *(accepted) ASME Intl. Design Engg. Tech. Conf.*

Kamimura, A.; Yoshida, E.; Murata, S.; Kurokawa, H.; Tomita, K.; and Kokaji, S. 2008. Distributed self-reconfiguration of m-tran iii modular robotic system. *Intl. J. of Rob.* 27(3-4):373–386.

Myerson, R. 1997. *Game Theory: Analysis of Conflict*. Cambridge, Massachusetts: Harvard University Press.

Puterman, M. 1995. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, USA.

Rahwan, T. 2007. *Algorithms for Coalition Formation in Multi-Agent Systems*. Ph.D. Dissertation, University of Southampton.

Rosa, M.; Goldstein, S.; Lee, P.; Campbell, J.; and Pillai, P. 2006. Scalable shape sculpturing via hole motions. In *IEEE Intl. Conf. Rob. and Auton.*, 1462–1468.

Sandholm, T.; Larson, K.; Andersson, M.; Shehory, O.; and Tohme, F. 1999. Coalition structure generation with worst case guarantees. *Artificial Intelligence* 111(1-2):209–238.

Stoy, K.; Brandt, D.; and Christensen, D. 2010. *Self-Reconfigurable Robots: An Introduction*. Cambridge, Massachusetts: The MIT Press.

Yim, M., and et al. 2007. Modular self-reconfigurable robot systems: Challenges and opportunities for the future. *IEEE Robotics and Automation Magazine* 14(1):43–53.