Dynamic Reconfiguration in Modular Robots using Graph Partitioning-based Coalitions

Prithviraj Dasgupta, Vladimir Ufimtsev
Computer Science Department
University of Nebraska, Omaha, USA
{pdasgupta, vufimtsev}@unomaha.edu

Carl Nelson, S. G. M. Hossain
Mechanical Engg. Department
University of Nebraska, Lincoln, USA
cnelson5@unl.edu, smgmamur@yahoo.com

ABSTRACT

We consider the problem of dynamic self-reconfiguration in a modular self-reconfigurable robot (MSR). Previous approaches to MSR self-reconfiguration solve this problem using algorithms that search for a goal configuration in the MSR's configuration space. In contrast, we model the self-reconfiguration problem as a constrained optimization problem that attempts to minimize the reconfiguration cost while achieving a desirable configuration. We formulate the MSR self-reconfiguration problem as finding the optimal coalition structure within a coalition game theoretic framework. To reduce the complexity of finding the optimal coalition structure, we represent the set of all robot modules as a fully-connected graph. Each robot module corresponds to a vertex of the graph and edge weights represent the utility of a pair of modules being in the same coalition (or, connected component). The value of a coalition structure is then defined as the sum of the weights of all edges that are completely within the same coalition in that coalition structure. We then use a graph partitioning technique to cluster the vertices (robot modules) in the constructed graph so that the obtained coalition structure has close to optimal value. The clustering algorithm has time complexity polynomial in the number of agents, n, and yields an $O(\log n)$ approximation. We have verified our technique experimentally for a variety of settings. Our results show that the graph clustering-based self-reconfiguration algorithm performs comparably with two other existing algorithms for determining optimal coalition structures. [1]

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Autonomous vehicles—*modular robots, dynamic reconfiguration*

General Terms

Algorithms

[1]This research has been performed as part of the ModRED project that is supported by a NASA EPSCoR grant.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Keywords

modular self-reconfigurable robots, dynamic reconfiguration, coalition game, graph-based clustering

1. INTRODUCTION

Over the past few years, modular self-reconfigurable robots (MSRs) have been proposed as an elegant, yet efficient way to build robots that are capable of maneuvering in tight spaces or unstructured terrain [18]. Structurally, an MSR is composed of functionally simple modules that are connected together into a certain formation. Each module is individually capable of performing very limited operations, but when connected with other modules, they can adapt their shape to form a single robot that can accomplish a complex task. In spite of the simple and inexpensive construction of an MSR's modules, and easy maneuverability, a principal challenge in MSRs is to solve the self-reconfiguration problem i.e. 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. As a motivating example, we consider a scenario where a set of robot modules are deployed individually, possibly scattered on the ground within communication range of each other, from an airborne vehicle. The objective of these individual modules is to autonomously determine suitable multi-module configurations, maneuver themselves to get within close proximity of each other, and, finally align and dock with each other to realize those configurations. This paper focuses on the computational aspects of the problem faced by the individual modules to determine their 'best' set of configurations that gives them an improved efficiency or value in performing their assigned task, while considering the costs in terms of energy expended to get in proximity of, and align and dock with each other to get into those configurations. 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 (CSG) 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, solving the CSG problem that deals with finding the *best* or optimal coalition in a coalition structure graph is known to be an NP-hard problem with a few existing heuristic solutions. To address these problems, in this paper we first develop a utility-based formulation for the costs corresponding to the dynamic reconfiguration problem in MSRs within a coalition game theoretic framework. Then we use a graph clustering algorithm to solve the CSG problem within this setting, using a polynomial time complexity and logarithmic approximation. To illustrate the operation of our MSR we have used the domain of robotic exploration of initially unknown environments. Our experimental results show that our graph clustering technique can be successfully used to dynamically self-reconfigure an MSR into different configurations.

2. 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 [4, 16]. An excellent overview of the state of the art MSRs and related techniques is given in [18]. 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 [3, 5] and control-based techniques [14]. 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 [9]. 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. Our work in this paper is targeted towards task-based reconfiguration techniques; we do not explicitly specify a goal configuration but allow the reconfiguration algorithm to select a new configuration that minimizes the reconfiguration cost.

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 [10, 13]. 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 [15] and Rahwan [11] have proposed anytime algorithms to find

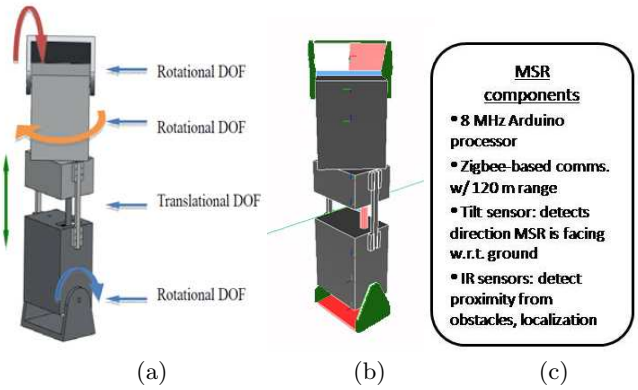

Figure 1: (a) CAD figure of a single module of the MSR. (b) A simulated version of the MSR inside Webots. (c) Major components of the MSR.

near-optimal solutions. In contrast to these works, we use a graph clustering-based approach to find the optimal coalition structure.

Weighted graph games were introduced in [8] as a specific case of coalition games where the set of agents is modelled as a vertex set of a graph, and the valuation function is calculated by summing the edge weights of the constructed graph. Coalition structure generation in graph games has recently received attention in [17], [1], and the problem was shown to be NP-complete. Our method uses the graph coalition game formulation and the graph clustering technique presented in [7] to obtain close to optimal coalition structures in the graph. The technique is based on a generalized version of correlation clustering [2] known as correlation clustering with partial information.

3. A NOVEL 4-DOF MODULAR ROBOT

We have used an MSR called ModRED [6] that is currently being developed by us, for implementing and testing the techniques in this paper. Unlike most other MSRs, it has 4 DOF (3 rotational and 1 translational); this allows each module to rotate along its long axis as well as extend along that same axis, as shown in Figure 1(a). This combination of DOF enables the MSR to achieve a greater variety of gaits to possibly maneuver itself out of tight spaces. A picture of the MSR, its simulated version within a robot simulator called 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 the MSR in fixed configuration is enabled through gait tables [16]. 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 MSR in different configurations using gait tables are available at http://cmantic.unomaha.edu/projects/modred/. The MSR ModRED can be configured into a chain struc-

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

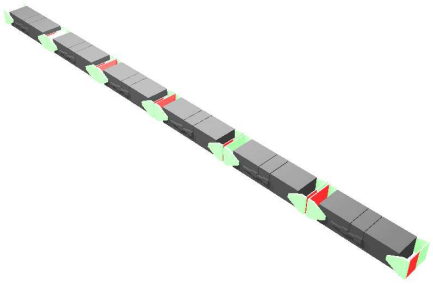

Figure 2: ModRED modules in a chain configuration

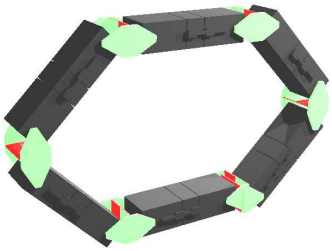

Figure 3: ModRED modules in a ring configuration

ture as is shown in Figure 2 as well as a ring structure as is shown in Figure 3 (images taken from Webots). When modules form these configurations the MSR performs more efficiently; it can move faster and it can overcome obstacles. In different configurations, the MSR uses different gaits which enable it to move faster and overcome more obstacles. In the chain structure the MSR can mimic the movement of a snake for movement or it can use its rotational degree of freedom to roll sideways, while in the ring structure the MSR can perform several "rolling" motions akin to that of a wheel.

While moving in a fixed configuration, if the MSR'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 graph clustering approach for finding the optimal coalition.

4. 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 (coalition structure) at time t, $\{A_i^t\}$, is defined as a set of exhaustive and disjoint partitions of A, i.e., $\cup_i A_i^t = A$ and $A_i \cap A_j = \emptyset$ for $i \neq j$. The i-th MSR (coalition) at time t is given by a set of ordered modules or agents, i.e.,

$$A_i^t = \{a_{i_1}^t, a_{i_2}^t, a_{i_3}^t, ..., a_{i_{|A_i^t|}}^t\} \quad (1)$$

Here, $a_{i_1}^t$ is the leading module of A_i^t, $a_{i_{|A_i^t|}}^t$ is the trailing or end module of the MSR and $\{a_{i_j}^t, a_{i_{j+1}}^t\}, j = 1...|A_i^t| - 1$ are the set of modules that are physically coupled together pairwise in a chain configuration using their end couplers. Using this definition, when A_i^t is a singleton, it represents a single module that is not coupled with any other modules.

Let $\Pi(A)$ be the set of all partitions of A and let $CS(A) = \{A_1, A_2, ..., A_k\} \in \Pi(A)$ denote a specific partition of A. [3] We define $V : \Pi(A) \rightarrow \mathbb{R}$, a value function that assigns each partition $CS(A) \in \Pi(A)$ a real number. Consider an MSR in configuration $CS_{old}(A) = \{A_1^{old}, A_2^{old}, ..., A_k^{old}\}$ that reconfigures to $CS_{new}(A) = \{A_1^{new}, A_2^{new}, ..., A_{k'}^{new}\}$. Note that k and k' may be different. Such reconfigurations can happen, for example, when an MSR gets stuck at an obstacle while navigating during an exploration task. The objective of the MSR is to get into a new configuration that lets it continue performing its assigned task while incurring the minimum reconfiguration cost. We parametrize the reconfiguration cost in the following manner. Let $cost_{CS_{old}(A) \rightarrow CS_{new}(A)}(a_i, a_j)$ denote the cost that will be incurred to couple modules a_i and a_j with each other (in the process of reconfiguration from $CS_{old}(A)$ to $CS_{new}(A)$). For simplicity we will write $cost_{old,new}(a_i, a_j)$. If these two modules remain in the same MSR after reconfiguration, this cost is 0. Otherwise, this cost is given by the sum of the costs of undocking the modules a_i and a_j respectively from their current MSRs, the cost of one of the modules, say a_j, moving to the vicinity of the other module a_i and the cost of aligning and docking the two modules, as described below:[4]

$$cost_{old,new}(a_i, a_j) = \begin{cases} 0, & \text{if } a_i, a_j \in A_{k_1}^{old} \\ & \text{and } a_i, a_j \in A_{k_2}^{new} \\ cost_{Undock}(a_i) & \\ + cost_{Undock}(a_j) & \\ + cost_{Crawl}(a_j, loc(a_i)) & \\ + cost_{AlignAndDock}(a_i, a_j), & \text{otherwise} \end{cases} \quad (2)$$

Within this framework, we define the MSR self-reconfiguration problem as the following:

Definition 1. **Modular Self Reconfiguration.** Given a partition $CS_{old}(A) = \{A_1^{old}, A_2^{old}, ..., A_k^{old}\}$, find a partition $CS_{new}(A) = \{A_1^{new}, A_2^{new}, ..., A_{k'}^{new}\}$ such that the following constraint is satisfied:

$$\max_{CS_{new}(A) \in \Pi(A)} V(CS_{new}(A)) - \sum_{a_i, a_j \in A} cost_{old,new}(a_i, a_j) \quad (3)$$

4.1 Coalition Game for MSR Self-Reconfiguration

We formulate the MSR self-reconfiguration problem as finding an optimal coalition structure using a coalition game theoretic framework. 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

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

[4] We assume that costs are symmetric, i.e., $cost_{old,new}(a_i, a_j) = cost_{old,new}(a_j, a_i)$.

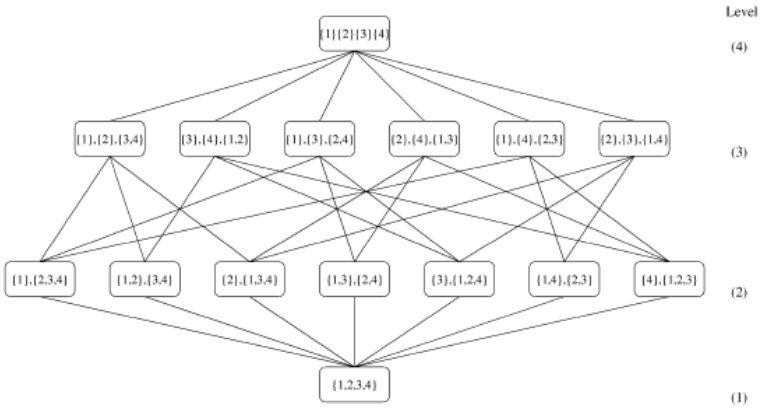

Figure 4: Coalition structure graph with 4 agents

self reconfiguration problem. We have used a popular representation of coalition games called characteristic function games (CFG) [10]. A CFG is defined by a pair of attributes (A, v), where A is the set of agents, and $v : 2^A \rightarrow \mathbb{R}$ 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 A agents. A *coalition structure* is an enumeration of the subsets S of A such that every agent appears exactly once in one of the subsets. For a set of agents A, let $\Pi(A)$ denote the set of coalition structures. For example, with $A = \{a_1, a_2, a_3\}$, $\Pi(A) = \{\{a_1\}\{a_2\}\{a_3\}, \{a_1\}\{a_2, a_3\}, \{a_2\}\{a_1, a_3\}, \{a_3\}\{a_1, a_2\}, \{a_1, a_2, a_3\}\}$. $\Pi(A)$ can be enumerated recursively as a coalition structure graph (CSG), as shown in Figure 4.1. Each coalition structure $CS(A) \in \Pi(A)$ 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 $CS(A)$ is associated with a value $V(CS(A))$ that is usually calculated by adding the values of each coalition within the coalition structure, i.e., $V(CS(A)) = \sum_{S \in CS(A)} v(S)$. For example, with 4 agents, for the coalition structure $\{a_1, a_2\}\{a_3\}\{a_4\}$, $V(\{a_1, a_2\}\{a_3\}\{a_4\}) = v(\{a_1, a_2\}) + v(a_3) + v(a_4)$. For the context of MSR reconfiguration, an agent a_i corresponds to a single MSR-module, a coalition S corresponds to an MSR A_j, while a coalition structure corresponds to a set of MSRs. To solve the modular 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 $CS^*(A) = arg \max_{CS(A) \in \Pi(a)} V(CS(A))$.

4.2 Graph-based Representation of CSG

Let $G = (A, E)$ denote a weighted, complete graph with its vertex set as the set of modules A, edge set as $E = \{(a_j, a_k) : \forall a_j, a_k \in A\}$ and edge weight function $w : E \rightarrow \mathbb{R}$ defined as:

$$w(e) = w(a_j, a_k) = Val - cost(a_j, a_k) \tag{4}$$

where Val is a fixed constant. As a simplification, we take the *cost* function to be symmetric so that $w(e) = w(e'), \forall e = (a_i, a_j), e' = (a_j, a_i) \in E$ and thus the graph can be treated as undirected. We define the sum of the edge weights in a coalition A_i to be the utility of the coalition, i.e.

$$v(A_i) = \sum_{\substack{e=(a_j,a_k): \\ a_j,a_k \in A_i, j \neq k}} w(e) \tag{5}$$

It is important to note that the edge weight function is not non-negative (which is a requirement of most graph clustering algorithms). Edges which have a positive weight correspond to a positive contribution to utility if the two modules were to join the same coalition, whereas negative edges will correspond to a decrease in utility if they were to join the same coalition. Also, with this definition, the utility of a singleton, i.e. a coalition consisting of only one module, is 0 since there are no edges within the coalition.

Graph-based MSR Reconfiguration Problem. Recall that $\Pi(A)$ is the set of all partitions of A i.e. the set of all possible non-overlapping coalition structures.

The utility of a *coalition structure* $CS(A) = \{A_1, A_2, ..., A_k\}$, is given by:

$$V(CS(A)) = \sum_{i=1}^{k} v(A_i) \tag{6}$$

Initially, A has a coalition structure consisting entirely of singletons i.e. each module is on its own and no coalitions of two or more modules have been formed. In this setting, the problem of finding an *optimal* coalition structure consists of clustering the graph $G = (A, E)$ into $CS^*(A) = \{A_1, A_2, ..., A_k\}$ such that:
$V(CS^*(A)) = \max_{CS(A) \in \Pi(A)} V(CS(A))$

4.3 Graph Clustering Approach for Coalition Formation

We will use the approach proposed in [7]. The *penalty* of a coalition structure $CS(A) = \{A_1, A_2, ..., A_k\}$ takes into account positive weighted edges *between* different coalitions in the structure and negative weighted edges *within* the same coalition in the structure and is defined to be:

$$Penalty(CS(A)) = Penalty_p(CS(A)) + Penalty_m(CS(A))$$

$$Penalty_p(CS(A)) = \sum_{\substack{e=(a_i,a_j):w(e)>0 \\ a_i \in A_{k_1}, a_j \in A_{k_2}, \\ k_1 \neq k_2}} |w(e)|$$

$$Penalty_m(CS(A)) = \sum_{\substack{e=(a_i,a_j):w(e)<0 \\ a_i,a_j \in A_{k_1}}} |w(e)|$$

The penalty of a coalition structure is equivalent to what is defined as the cost of a clustering in [7]. We use the term "penalty" so there is no confusion with the *cost* function which was defined in Equation 2. To obtain a coalition structure with close to optimal utility, we are interested in maximizing the sum of edge weights that are within coalitions in the structure. That is, it is beneficial to have modules (vertices) that have a positive edge between them to be in the same coalition, and to have modules that have a negative edge to be in different coalitions. Notice that by reducing the total weight of negative edges within coalitions, and total weight of positive edges between coalitions, the utility of the coalition is increased. By minimizing the penalty we are thus minimizing the absolute total weight of positive edges between coalitions and absolute total weight of negative edges completely within coalitions, and therefore increasing the utility of the coalition structure. In fact, as we will show, minimizing the penalty of a coalition structure is equivalent to maximizing its utility.

As in [7], for each pair of modules (vertices) i.e. for each edge $(a_i, a_j) \in E$, we introduce binary variables $x_{a_i a_j} \in \{0,1\}$ for a clustering $CS(A) = \{A_1, A_2, ..., A_k\}$ such that $x_{a_i a_j} = 0 \leftrightarrow \exists A_l \in CS(A) : a_i, a_j \in A_l$ (the two modules are in the same coalition) and $x_{a_i a_j} = 1 \leftrightarrow \exists A_{k_1}, A_{k_2} \in CS(A), k_1 \neq k_2 : a_i \in A_{k_1}, a_j \in A_{k_2}$ (the two modules are in different coalitions). Notice that $1 - x_{a_i,a_j} = 0$ iff a_i and a_j are in different coalitions and $1 - x_{a_i,a_j} = 1$ iff a_i and a_j are in the same coalition. We will use the abbreviation x_e for $x_{a_i a_j}$ where $e = (a_i, a_j) \in E$. Formulating $Penalty(CS(A))$ using these binary variables, we need non-negative constants:

$$m_e = \begin{cases} |w(e)| & \text{if } w(e) < 0 \\ 0 & \text{if } w(e) \geq 0 \end{cases}$$

$$p_e = \begin{cases} |w(e)| & \text{if } w(e) > 0 \\ 0 & \text{if } w(e) \leq 0 \end{cases}$$

So $Penalty(CS(A))$ becomes:

$$Penalty(CS(A)) = \sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e) \qquad (7)$$

We wish to find a coalition structure with minimal cost. This is equivalent to finding the structure with optimal utility as we now show.

Proposition 1. A coalition structure with minimal penalty will have a maximal (optimal) utility.

Proof: From Equation 7 we have that the penalty of a coalition structure $CS(A)$ is:

$$Penalty(CS(A)) = \sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e)$$

Using the same notation, we have that the utility of a coalition structure from Equations 5, 6 is:

$$V(CS(A)) = \sum_{i=1}^{k} v(A_i) = \sum_{i=1}^{k} \sum_{\substack{e=(a_j,a_k): \\ a_j,a_k \in A_i, j \neq k}} w(e) =$$
$$\sum_{e \in E} p_e(1 - x_e) - \sum_{e \in E} m_e(1 - x_e) =$$
$$\sum_{e \in E} p_e - \left(\sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e) \right) =$$
$$\sum_{e \in E} p_e - Penalty(CS(A))$$

Since $\sum_{e \in E} p_e$ is a constant (the variables are the x_e's), then minimizing $Penalty(CS(A))$ is equivalent to maximizing $V(CS(A))$ and thus the coalition with minimal penalty will have the maximal (optimal) utility.

□

Notice that $Penalty(CS(A)) \geq 0, \forall CS(A) \in \Pi(A)$ and to minimize the Penalty using the above formulation, it would suffice to set $x_e = 0$ whenever $p_e > 0$ and $x_e = 1$ whenever $m_e > 0$ so that $Penalty = 0$, however this will not necessarily correspond to a valid coalition structure since the x_e variables are not independent. As noted in [7], an assignment of values $\{0,1\}$ to the variables x_e corresponds to a valid coalition structure (clustering) if the variables satisfy the triangle inequality, that is $\forall a_i, a_j, a_k \in A, i \neq j \neq k$, $x_{a_i,a_j} + x_{a_j,a_k} \geq x_{a_i,a_k}$. The problem can therefore be formulated as a 0-1 integer linear program:

min: $\sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e)$

constraints:
$x_{a_i,a_j} \in \{0,1\}, \forall a_i, a_j \in A, i \neq j$
$x_{a_i,a_j} + x_{a_j,a_k} \geq x_{a_i,a_k}, \forall a_i, a_j, a_k \in A, i \neq j \neq k$
$x_{a_i,a_j} = x_{a_j,a_i} \forall a_i, a_j \in A, i \neq j$

As is known, 0-1 integer linear programming is NP-complete, so the problem is relaxed to a linear program [7]:

min: $\sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e)$

constraints:
$x_{a_i,a_j} \in [0,1], \forall a_i, a_j \in A, i \neq j$
$x_{a_i,a_j} + x_{a_j,a_k} \geq x_{a_i,a_k}, \forall a_i, a_j, a_k \in A, i \neq j \neq k$
$x_{a_i,a_j} = x_{a_j,a_i} \forall a_i, a_j \in A, i \neq j$

The problem can now be solved in polynomial time, though it may yield a fractional solution i.e. the variables x_e are in the interval $[0,1]$, and are not necessarily binary. In this case the authors in [7] propose a rounding algorithm based on region growing to obtain a valid approximate solution i.e. obtain a valid assignment of 0's and 1's to the x_e variables which will yield a close to minimal penalty. The whole algorithm runs in polynomial time (polynomial in the number of variables) and is a $O(\log n)$ approximation. The number of variables is simply the number of edges in

the graph. In our case, we have a complete graph so that $|E| = \binom{|A|}{2} = \binom{n}{2} = \frac{n(n-1)}{2}$ and so the algorithm will run in time polynomial in the number of modules (vertices) n. The algorithm is outlined as follows:

1. Coordinates of all agents in A are specified. Parameter Val is specified.

2. $\forall a_i, a_j \in A$, $w(a_i, a_j)$ is calculated using Equations 2 and 4.

3. Objective function given by Equation 7 is formulated and constraints are set.

4. Linear programming is used to obtain a solution to the optimization problem.

5. If the solution is 0-1 integer then a valid coalition structure has been found. Else if the solution is fractional, the region growing rounding algorithm [7] is used to obtain a valid approximate solution.

5. EXPERIMENTAL RESULTS

We have implemented the described algorithm for obtaining optimal coalition structures using the mixed integer linear programming solver LPsolve version 5.5.2.0. In this section we present results on simulations for agent set sizes from 3 to 43. For agent set sizes between 3 and 12, we were able to perform an exhaustive search on the space of all coalition structures to find the actual optimal coalition structure and compare it with the structure obtained using the graph clustering linear programming model. For higher agent set sizes the exhaustive search (complexity $O(n^n)$) becomes prohibitive, as do the algorithms for the general CSG problem [11, 15]. The reason we are able to outperform those methods is that our formulation is a restricted case i.e. we are restricting the problem to a weighted graph where coalition utilities are calculated by summing pairwise utilities (edge weights).

For each of the agent set sizes $n = \{3, 4, ..., 43\}$, we used a grid size of $(n+4) \times (n+4)$ and generated random integer coordinates for each of the agents i.e. for each agent $a_i \in A, 1 \leq i \leq n$, we assigned coordinates (x_{a_i}, y_{a_i}) where x_{a_i} is randomly generated from $\{0, 1, ..., n+3\}$ and y_{a_i} is randomly generated from $\{0, 1, ..., n+3\}$. We generated 30 random arrangements of n agents in an $(n+4) \times (n+4)$ grid for each value of n from 3 to 12, and 10 random arrangements for n from 13 to 43. The edge weights were calculated using $w(e) = Val - cost(a_j, a_k)$, and for simplification we took $cost(a_j, a_k) = d(a_j, a_k) + cost_{AlignAndDock}$ where $d(a_j, a_k)$ is the Euclidean distance between agents a_j, a_k in the grid and $cost_{AlignAndDock} = 1$ was a constant representing the cost of alignment and docking of two modules (agents). We set $Val = \frac{n}{2} + 3$ for each different value of n. Val was chosen so that for each edge (a_i, a_j), the weight is $w(e) = Val - cost(a_j, a_k) = \frac{n}{2} + 3 - d(a_j, a_k) - 1 = \frac{n+4}{2} - d(a_j, a_k)$ meaning if modules a_i, a_j are within a distance of $\frac{n+4}{2}$ (half the width of the grid) then their edge weight (utility) will be positive, and if they are at least half a grid width apart then they will have a negative edge weight. For singleton coalitions we used the utility of 0. Notice that in the graph formulation, singleton coalitions have no edges contained and hence do not contribute anything to the coalition utility[5]. A zero utility for singletons also indicates that a coalition consisting of a single module will not provide any contribution to the total utility of the coalition structure.

Such a scenario arises in a practical setting where the modules are deployed in an unknown environment as singletons. Deploying a configured MSR is harder than deploying single modules separately. Each of the individual modules may be dropped from an aircraft/spacecraft with a parachute and thus will be scattered in the ground environment once they land. Our goal is then to find the optimal coalition structure so that the modules can form into larger MSRs and proceed with exploration.

After randomly generating the test cases for each agent set size and calculating the edge weights, we formulated the objective function (the penalty of a coalition structure) to minimize and the constraints, then used LPsolve to produce a valid solution. For agent set sizes up to 43, using the default settings in LPsolve, all solutions to the randomly generated test cases were 0-1 integer i.e. no rounding is required. In this case an exact solution is obtained to the 0-1 integer linear programming model and thus it is an optimal coalition structure. As empirical evidence of this we took the actual optimal utility from the exhaustive search and compared to the value obtained using LPsolve.

Table 1 shows the mean ratio (averaged over the 30 test cases for each agent set size) of the utility of the coalition structure we found to the utility of the optimal coalition structure. Mean runtime (average over each of the 30 test cases) is also displayed. For implementation we used a desktop PC (Intel Core i7 - 960 3.20GHz, 12GB DDR3 SDRAM)

No. of Agents	Mean Ratio to Optimal Utility	Mean Runtime (secs)
3	1	0.004667
4	1	0.004733
5	1	0.004833
6	1	0.005
7	1	0.005233
8	1	0.005733
9	1	0.006367
10	1	0.007233
11	1	0.008567
12	1	0.01127

Table 1. Mean Ratio of Utilities for Coalition Structures Obtained to Optimal Coalition Structures

From Table 1, we see that the method we implemented is able to find optimal coalition structures for the given number of agents. In fact, in all of the 30 cases for each agent size, optimal coalition structures were found.

Figure 5 shows running times averaged over 10 test cases for each value of agent set size from 3 to 43. The algorithm was implemented for agents set sizes 13 to 43 on the same machine. Notice that even for 43 agents, the running time is approximately 5 seconds. Running algorithms which explore the space of all coalition structures for such numbers

[5]While realistically, singleton coalitions should have a value, we do not consider this in our formulation. Adding vertex weights to the formulation is a possible way to account for singleton coalitions.

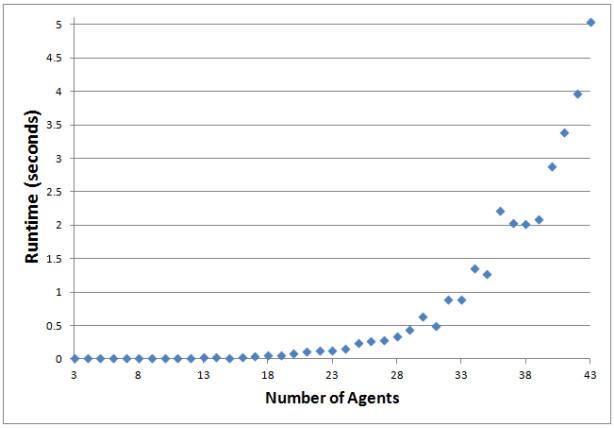

Figure 5: Average running times for various agent set sizes

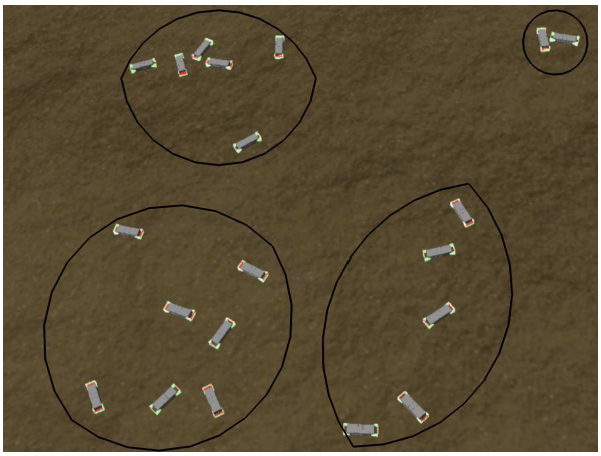

Figure 6: Initial locations of 20 ModRED modules randomly scattered within an environment (snapshot from Webots)

of agents is prohibitive. The algorithms are all bounded below by $\Omega(2^n)$ since the utility of every possible subset of A has to be obtained. While the algorithms [15], [11] are applicable to any coalition game i.e. any valuation function, the method implemented by us is only applicable to the graph coalition game case where the valuation function is calculated by summing pairwise utilities.

Figure 6 shows the initial random arrangement of 20 ModRED modules in a 24×24 grid (snapshot taken from Webots). Each module is assumed to be in its own singleton coalition, edge weights are calculated using equation 5 with $Val = 15$, $cost = 1$, and Euclidean distances between modules. The coalition structure obtained using the graph clustering method with the given parameters is displayed (modules in the same coalition are circled). The clustering method naturally groups close modules into the same cluster (since with constant *cost*, edge weights are primarily influenced by distance). Once the coalition structure is determined, the modules are instructed to form into their respective coalitions and configure into chains as is illustrated in Figure 7.

For agent set sizes larger than 43, LPsolve produced fractional solutions i.e. the edge variables x_{a_i,a_j} were values in

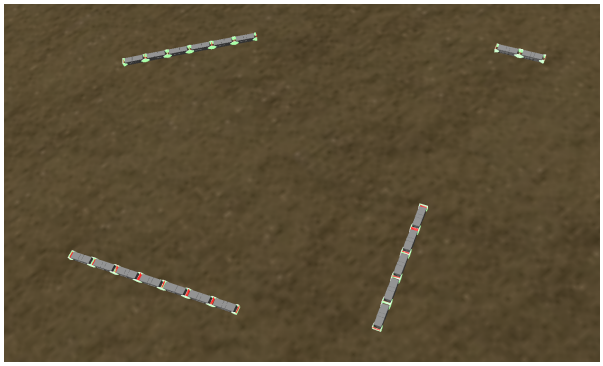

Figure 7: ModRED modules form the specified coalitions and configure into four different chain formations (snapshot from Webots)

the interval (0,1). In this case, the region growing rounding algorithm in [7] is implemented to obtain $O(\log n)$-approximations to the optimal coalition structure. Additional time is required for this procedure but it should be noted that it runs in polynomial time and therefore does not change the fact that the overall algorithm runs in polynomial time.

6. CONCLUSION AND FUTURE WORK

We have formulated the CSG problem for our setting of MSR reconfiguration as a graph coalition formation game. Current state of the art algorithms for general coalition formation are all bounded below by $\Omega(2^n)$ [15], [11] and for a guaranteed optimal solution, the worst case running time is $O(n^n)$. In our formulation, although it is a specific case with a graph representation, there is no longer a $\Omega(2^n)$ lower bound and algorithms exist [7] which guarantee a $O(\log n)$ approximation and run in polynomial time. In this setting, coalition formation for large sets of agents becomes feasible.

In general, solving the 0-1 integer linear programming problem (which is part of the graph clustering technique) is NP-Complete, but when the problem is relaxed to a general linear programming problem, solutions can be found in polynomial time. Fractional solutions do not represent a valid coalition structure and a $O(\log n)$ approximation polynomial time algorithm is used to obtain a valid coalition structure. In the simulation results presented, we did not have to implement the region growing rounding algorithm in [7] since we obtained 0-1 integer solutions and thus optimal coalition structures. As the agent set size grows larger, the solutions obtained in the linear programming part of the algorithm are fractional and region growing has to be applied to obtain valid coalition structures. We plan to extend our work to include the rounding algorithm so that the problem can be approximately solved for larger agent set sizes and compare it with more recent coalition structure search algorithms [12].

Also, our current formulation is able to produce a close to optimal coalition structure when reconfiguring from the coalition structure in which each module is in a coalition on its own i.e. each coalition is a singleton. We plan to extend the approach so that we can reconfigure from any given initial configuration. Assigning tasks for the coalitions is another extension we hope to explore. In this scenario

each coalition is assigned a task and constraints are set so that each coalition is able to solve the assigned task. In the case when not all tasks can be solved, reconfiguration has to be performed to obtain a coalition structure which will meet the task needs. Implementing our approach in the physical world to the MSR ModRED requires that we take into account uncertainty since sensor noise becomes a key issue. Extending our formulation by using stochastic edge weights is one approach to tackling the issue of uncertainty in the graph coalition formation problem.

7. REFERENCES

[1] Y. Bachrach, P. Kohli, V. Kolmogorov, and M. Zadimoghaddam. Optimal coalition structures in graph games. *arXiv:1108.5248v1 [cs.GT]*, 2011.

[2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

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

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

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

[6] K. Chu, S. G. M. Hossain, and C. Nelson. Design of a four-dof modular self-reconfigurable robot with novel gaits. In *ASME International Design Engineering Technical Conference*, pages DETC2011–47746, Washington, D.C., 2011.

[7] E. Demaine and N. Immorlica. Correlation clustering with partial information. *Lecture Notes in Computer Science*, 2764:71–80, 2003.

[8] X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics and Operations Research*, 19(2):257–266, 1994.

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

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

[11] T. Rahwan. *Algorithms for Coalition Formation in Multi-Agent Systems*. PhD thesis, University of Southampton, 2007.

[12] T. Rahwan, S. Ramchurn, A. Giovannucci, and N. Jennings. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, 34:521–567, 2009.

[13] D. Ray. *A Game-Theoretic Perspective on Coalition Formation (1st ed.)*. Oxford University Press, USA, 2008.

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

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

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

[17] T. Voice, M. Poulakarov, and N. Jennings. Graph coalition structure generation. *arXiv:1102.1747v1 [cs.DS]*, 2011.

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