A Distributed Reconfiguration Planning Algorithm for Modular Robots

Chao Liu, Michael Whitzer, and Mark Yim

***Abstract*—Self-reconfigurable modular robots are usually composed of multiple modules with uniform docking interfaces that can be transformed into different configurations by themselves. The reconfiguration planning problem is finding what sequence of reconfiguration actions are required for one arrangement of modules to transform into another. We present a novel reconfiguration planning algorithm for the SMORES form of modular robots. The algorithm compares the initial configuration with the goal configuration efficiently. The reconfiguration actions can be executed in a distributed manner so that each module can efficiently finish its reconfiguration task which results in a global reconfiguration for the system. In the end, the algorithm is demonstrated on the SMORES-EP self-reconfigurable modular robot hardware and some reconfiguration task examples are provided.**

***Index Terms*—Cellular and Modular Robots, Motion and Path Planning, Path Planning for Multiple Mobile Robots or Agents.**

I. Introduction

SELF-RECONFIGURABLE modular robots are usually composed of a small set of building blocks, with uniform docking interfaces that allow the transfer of mechanical forces and moments, electrical power, and communication throughout the robot [1]. These systems are able to adapt to many different activities, handle hardware and software failures by reconfiguring themselves [2]. A fundamental problem to achieve these goals is known as the *self-reconfiguration planning* problem.

The self-reconfiguration ability enables a modular system to change the arrangement of modules from one arbitrary configuration to another. While the reconfiguration planning problem is well defined, the problem is usually difficult to solve because of the physical constraints particular to each self-reconfigurable robot system. For example, the modules are usually designed to be simple to manufacture and low-cost so that each module only has a limited number of actuators and connectors. This results in limited motion ability and often complex system constraints. In addition, the number of possible arrangements for a cluster of modular robots, grows exponentially with the number of modules which makes the planning problem intractable to find optimal solutions with naïve brute-force techniques.

The majority of self-reconfigurable robots are typically divided into three main types: chain-type, lattice-type and mobile-types with some models that are hybrid between these types. Lattice-type self-reconfigurable robots sit nominally on a lattice and reconfigure between neighboring lattice positions. There are already many reconfiguration planning algorithms for lattice-type robots, such as [3] [4] [5] [6]. However, it is difficult for lattice-type modular robots to generate some dynamic locomotion and generic manipulation.

In contrast, chain-type self-reconfigurable robots are particularly well suited for locomotion and manipulation. Numerous chain-type modular robotic systems have been developed in the past few years, such as PolyBot [7], M-TRAN [8], SuperBot [9], and CKBot [10]. Self-reconfiguration with this type of robots is often a difficult and time consuming task [2].

The third type, mobile-type robots, use the environment to move between modules as they reconfigure. Self-reconfigurable systems that have used mobile-type reconfiguration include Millibots [11], Swarm-bots [12], Planar Catoms [13], and Kilobots [14]. In all of these cases, the systems are composed of large nunmbers of identical robots that can move around on a flat ground and connect together to form different planar shapes.

The system we use, SMORES [15], is a hybrid-type that can achieve all three types of reconfiguration. However, in this work we will focus on the mobile style of reconfiguration. For self-reconfiguration planning, graph representations of modular robot configurations are used and an efficient algorithm to find reconfiguration actions is developed in such a way that modules can act in a reasonable order to achieve the goal configuration efficiently.

The paper is organized as follows. Sec. II reviews relevant and previous work. Sec. III introduces the hardware platform. Sec. IV introduces some fundamental parts and concepts. The algorithm is presented in Sec. V and some experiments and results are shown in Sec. VI. Finally, Sec. VII talks about the conclusions and future work.

II. Related Work

The reconfiguration planning problem for modular robots has been addressed to some extent by some research work. Casal and Yim [16] firstly published a divide-and-conquer approach for this problem. Two algorithms were presented which are based on a wireframe depiction of modular robot configurations and a substructure set resulting in a hierarchy construction of initial and goal configurations. However, neither of them are parallel algorithms and many unnecessary motions are involved. Nelson [17] used graphs to represent modular robot configurations and the reconfiguration planning is generated by updating the difference matrix. The PCA

Manuscript received: February, 24th, 2019; Revised June, 10th, 2019; Accepted July, 1st, 2019.

This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by NSF grant numbers CNS-1329620 and CNS-1329692.

The authors are with GRASP Lab and Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104, USA {chaoliu, mwhitzer, yim}@seas.upenn.edu

Digital Object Identifier (DOI): 10.1109/LRA.2019.2930432.

method and weighted bipartite graph assignment solutions are used to determine the optimal graph matching. Both [16] and [17] only consider single-DoF modules and multiple ways of connections between two connectors are also not considered.

Payne et al. [18] presented an algorithm for chain-type modular robots but limited to reconfigure from an "I" shape to a "T" shape. Asadpour et al. [19] developed an algorithm for chain-type self-reconfigurable robots using guided search methods with a heuristic function which is the combination of a similarity metric and a depth variable between two configurations, and the graph signature is computed for an isomorphism test. However, this method is computationally slow because the search space of configurations is extremely large. Hou and Shen [20] presented configuration string to represent a robot configuration. Common and different substructures are detected and all different substructures can be reconfigured into an intermediate structure and then into goal substructures. However, the way to compare two configurations is with respect to the center of graphs so that it is very likely to not find common substructures, resulting in many redundant reconfiguration steps. A graph-based optimal reconfiguration planning approach [21] was developed later. The reconfiguration problem is converted into a distributed constraint optimization problem (DCOP) which is solved by existing DCOP algorithms whereas solving DCOP takes exponential time. Thus a greedy algorithm was introduced to solve the configuration matching problem but the optimal solution is not guaranteed. All of these works require that all modules are connected during the reconfiguration process due to the limited motion capabilities of their hardware.

A configuration recognition algorithm using distributed information for modular robots was presented in [22]. A graph representation for modular robots was presented, including the definition of root module and connections. A matching and mapping algorithm is developed for configuration recognition by computing the maximum common subconfiguration (MCS) with respect to root modules. This paper focuses on reconfiguration planning for modular robots which is based on these fundamental concepts and the bottom-up algorithm to compute MCS from [22]. An efficient configuration decomposition algorithm is developed and a virtual module operation is presented for further configuration decomposition until all modules are mapped. Then distributed reconfiguration actions can be derived from leaves to roots to ensure the locomotion ability of the module involved.

III. Hardware Platform

SMORES-EP (**S**elf-assembly **MO**dular **R**obot for **E**xtreme **S**hape-shifting) is a modular robotic system first published in [15]. SMORES-EP is the current version of the system where EP refers to the Electro-Permanent magnets the module uses for its connector [23]. [24] [25] have shown the ability and applications of this modular robotic system and some simple reconfiguration tasks were demonstrated.

Each module has four active rotational degrees of freedom (pan, tilt and left/right wheels) and four connectors which are

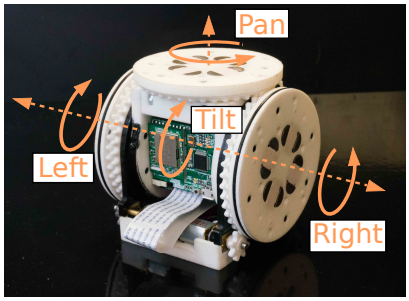
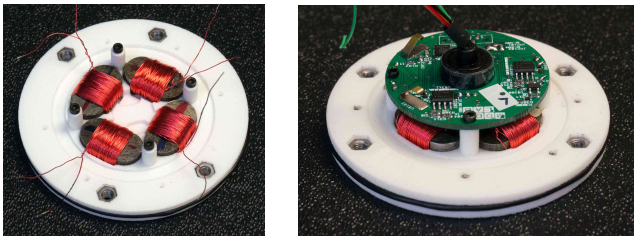


Fig. 1. A SMORES-EP module with four active rotational degrees of freedom and four connectors using an array of electro-permanent (EP) magnets.



(a) (b)

Fig. 2. (a) Internal view of magnets in EP-Face. (b) Internal view of EP-Face with circuit board.

equipped with an array of electro-permanent (EP) magnets as illustrated in Fig. 1. These four degrees of freedom are named *LEFT DoF*, *RIGHT DoF*, *PAN DoF* and *TILT DoF* for convenience. In particular, LEFT DoF, RIGHT DoF and PAN DoF can continuously rotate (no angular limits on rotation) to produce a twist motion of docking ports relative to the rest of the module, and TILT DoF is limited to $\pm 90^\circ$ to produce a bending joint. LEFT DoF and RIGHT DoF can be used as driving wheels which allows differential drive locomotion of individual modules.

A SMORES-EP module can be considered as a cube with four docking ports named *LEFT Face*, *RIGHT Face*, *TOP Face* and *BOTTOM Face* for convenience. Each face of the module can form a strong connection with other modules, or with metal objects by the use of four EP magnets arranged in a ring, with south poles counterclockwise of north shown in Fig. 2. The ring arrangement of the magnets makes the connector hermaphroditic, and able to connect in four possible configurations. Also connected EP-Faces are able to exchange data through the magnetic coupling of connected EP-magnets which are capable of UART serial communication [23].

IV. Preliminaries

The reconfiguration planning problem can be defined as finding the sequence of actions to convert an arbitrary configuration to another configuration. Graphs are concise representations for modular robot configurations and readily allow application of techniques from graph theory [26]. In particular, a modular robot configuration can be represented as a graph and each vertex of the graph represents a module and each edge of the graph represents the connection between two modules. Let $G = (V, E)$ be an undirected graph, where V

is the set of vertices of G representing the modules while E is the set of edges of G representing the connections among modules.

Graphs with only one path between each pair of vertices are *trees*. Any acyclic graph is a tree. It is convenient to treat the configurations as trees with manipulations on that tree to obtain different configurations. If the initial and/or goal configuration has loops, they can be converted into an acyclic configuration by running a spanning tree algorithm. Therefore, this work only focuses on configurations whose graphs are acyclic.

Once a tree $G = (V, E)$ is rooted with respect to a vertex $\tau \in V$, the parent of a vertex $v \in V$ is the vertex connected to it on the path to τ which is unique except for τ, and the child of a vertex $v \in V$ is a vertex of which v is the parent. The configuration of a modular robot cluster is represented as a rooted tree and the root has to be selected as the center of its graph defined in [27]. A linear-time algorithm to find the root of a given robot configuration is shown in [22]. The *degree* of a vertex is the number of edges incident to the vertex and a vertex of degree 1 is a leaf in the graph. The *height* of a vertex in a rooted tree is the length of the longest path (away from the root) to a leaf from that vertex and the height of the root is the height of the tree.

A modular robot module usually has multiple connectors and there may also be multiple ways to connect them. For each connection between two modules, the involved faces and orientations are meant to be considered.

***Definition** 1:* A connection between module u's connector U_Con and module v's connector V_Con with orientation Ori is defined as

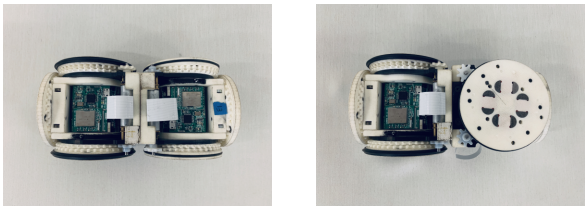
$$\begin{aligned}\text{connect}(u, v) =&\{\text{Face} : \text{U\_Con}, \text{Face2Con} : \text{V\_Con},\\ &\text{Orientation} : \text{Ori}\}\end{aligned} \tag{1}$$

from module u's point of view and

$$\begin{aligned}\text{connect}(v, u) =&\{\text{Face} : \text{V\_Con}, \text{Face2Con} : \text{U\_Con},\\ &\text{Orientation} : \text{Ori}\}\end{aligned} \tag{2}$$

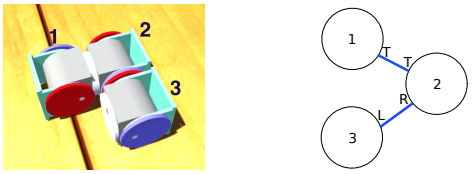
from module v's point of view.

Each connection has three attributes: *Face*, *Face2Con* and *Orientation*. For different designs of modular robots, some seemingly different connections are actually equivalent. In a SMORES-EP configuration, the connections among LEFT Face, RIGHT Face and TOP Face with different *Orientations* are actually equivalent. In contrast, for connections between two BOTTOM Faces which cannot rotate, the orientation needs to be considered ($\text{Orientation} \in [0, 1]$) shown in Fig 3. The *Orientation* attribute can affect the configuration kinematics. In addition, the module is bilaterally symmetric, namely LEFT Face or LEFT DoF is a mirror image of the RIGHT Face or RIGHT DoF, so the connections between LEFT Face and other docking ports are equivalent to the connections between RIGHT Face and the same mated docking port, and all possible connections by LEFT Face and RIGHT Face are equivalent.



(a) (b)

Fig. 3. Connection between two BOTTOM Faces: (a) Orientation is 0 and (b) Orientation is 1.



(a) (b)

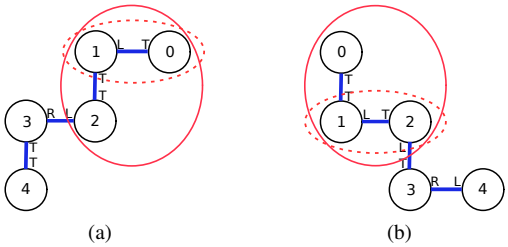
Fig. 4. (a) Three-Module Configuration and (b) Three-Module Configuration Graph.

A three-module SMORES-EP configuration is shown in Fig. 4. Here, the connections can be expressed as:

$$\begin{aligned}\text{connect}(1, 2) =&\{\text{Face} : \text{TOP Face},\\ &\text{Face2Con} : \text{TOP Face}, \text{Orientation} : \mathit{Null}\}\\ \text{connect}(2, 3) =&\{\text{Face} : \text{RIGHT Face},\\ &\text{Face2Con} : \text{LEFT Face}, \text{Orientation} : \mathit{Null}\}\end{aligned}$$

and $\text{connect}(2, 1)$ and $\text{connect}(3, 2)$ are similar. In addition, for this simple configuration, the root module is Module 2.

Given two modular robot configurations $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a *common subconfiguration* is defined in [22], as well as a *maximum common subconfiguration with respect to* $v_1 \in V_1$ *and* $v_2 \in V_2$ (denoted as $\text{MCS}(v_1, v_2)$). An example is shown in Fig. 5. Given two graphs G_1 and G_2 rooted with respect to $\tau_1 \in V_1$ and $\tau_2 \in V_2$ respectively, for module $v_1 \in V_1$ and $v_2 \in V_2$, we can construct a common subconfiguration $\{G_1', G_2'\}$ where $V_1' = \{v_1, \hat{v}_1\}$, $V_2' = \{v_2, \hat{v}_2\}$ under a subconfiguration mapping $f' : V_1' \to V_2'$ such that $f'(v_1) = v_2$ and $f'(\hat{v}_1) = \hat{v}_2$ if and only if $\text{connect}(v_1, \hat{v}_1) \cong \text{connect}(v_2, \hat{v}_2)$ which is called the



(a) (b)

Fig. 5. Two SMORES-EP configurations are shown. The subgraphs of (a) and (b) circled by "- -" is an example of common subconfiguration with mapping $1 \to 1$ and $0 \to 2$. The subgraphs of (a) and (b) circled by "—" is $\text{MCS}(1, 1)$ with mapping $1 \to 1$, $2 \to 0$ and $0 \to 2$.

feasibility rule. This feasibility rule can be used to find $\text{MCS}(v_1, v_2)\ \forall v_1 \in V_1$ and $\forall v_2 \in V_2$.

V. Reconfiguration Planning Algorithm

The self-reconfiguration planning problem can be stated as: Given an arbitrary initial configuration and goal configuration, find the actions required for the system to transform the initial configuration to the goal configuration. Different atomic reconfiguration actions can be defined for different modular robotic systems. Usually there are two atomic reconfiguration actions: *Docking* and *Undocking*. *Docking* means connecting two connectors and *Undocking* means disconnecting an existing connection. Different modular robots have different procedures to execute these two atomic actions. In particular, for SMORES-EP modular robotic system, a *Docking* action requires two modules to move their involved connectors to be in proximity, align these two faces and activate all corresponding magnets. Similarly, an *Undocking* action requires two modules to deactivate all magnets of both involved connectors.

An outline of the reconfiguration planning algorithm follows. The first step is to find the root module of the initial configuration as in [22], and then figure out the root module of the goal configuration. From here, a novel way to decompose the initial configuration and the goal configuration into multiple subconfigurations can be applied. These subconfigurations can then be mapped between initial and goal configurations. This mapping is computed by iteratively adding virtual modules and virtual connections to these subconfigurations. After the connection and disconnection actions are determined, the locomotion plans for modules to move to their required positions to connect can then be implemented with standard 2D path planning approaches. Commonly used nomenclature is listed in the following table.

G_i	Initial Configuration
G_g	Goal Configuration
$\bar{G}_i$	Initial Subconfiguration in MCS
$\bar{G}_g$	Goal Subconfiguration in MCS
$\hat{G}_i$	Initial Subconfiguration not in MCS
$\hat{G}_g$	Goal Subconfiguration not in MCS
G'_i	Initial Configuration after Virtual Module Operation
G'_g	Goal Configuration after Virtual Module Operation
τ_i	Initial Configuration Root Module
τ_g	Goal Configuration Root Module
τ_i^α	αth Subconfiguration Root of G_i
τ_g^β	βth Subconfiguration Root of G_g
$\mathcal{M}$	Virtual Module

A. Configuration Decomposition

Given the initial and goal configurations, we first decompose them into multiple subconfigurations. For efficiency, we want the decomposition that shares the most connections (edges in the graph) between initial and goal configurations. Reconfiguration actions (*Docking* and *Undocking*) are usually hard to execute and also time-consuming. Hence, one goal of the algorithm is to minimize the number of Docking and Undocking actions. In addition, it is hard to change the height of a vertex in the graph, for example, moving a module which is a leaf vertex of a configuration to a position close to the root requires more actions, so minimizing these changes will make the physical reconfiguration more efficient as well.

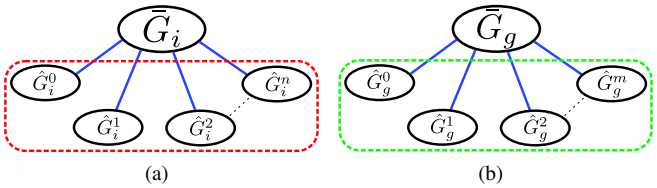


Fig. 6. (a) Configuration decomposition for $G_i = (V_i, E_i)$ and the subconfiguration encircled by "- -" is $\hat{G}_i = (\hat{V}_i, \hat{E}_i)$ and (b) configuration decomposition for $G_g = (V_g, E_g)$ and the subconfiguration encircled by "- -" is $\hat{G}_g = (\hat{V}_g, \hat{E}_g)$.

Given any two modular robot configurations $G_i = (V_i, E_i)$ and $G_g = (V_g, E_g)$, their root modules τ_i and τ_g can be computed in $\mathcal{O}(|V_i|)$ or $\mathcal{O}(|V_g|)$ respectively, then $\text{MCS}(\tau_i, \tau_g)$ under mapping $f : \bar{V}_i \to \bar{V}_g$ where $\bar{V}_i \subseteq V_i$ and $\bar{V}_g \subseteq V_g$ can be computed efficiently. These two subconfigurations $\bar{G}_i = (\bar{V}_i, \bar{E}_i)$ and $\bar{G}_g = (\bar{V}_g, \bar{E}_g)$ contained in $\text{MCS}(\tau_i, \tau_g)$ are isomorphic so that there is no need to reconfigure these modules. If subtracting subconfigurations $\bar{G}_i = (\bar{V}_i, \bar{E}_i)$ from $G_i = (V_i, E_i)$ without keeping boundaries, a graph $\hat{G}_i = (\hat{V}_i, \hat{E}_i)$ composed of multiple unconnected subgraphs is generated. Similar operations can be applied to the goal configuration $G_g = (V_g, E_g)$ to generate $\hat{G}_g = (\hat{V}_g, \hat{E}_g)$. The process is shown in Fig. 6 where $\hat{G}_i = \{\hat{G}_i^\alpha = (\hat{V}_i^\alpha, \hat{E}_i^\alpha) | \alpha = 0, 1, 2, \cdots, n\}$ and $\hat{G}_g = \{\hat{G}_g^\beta = (\hat{V}_g^\beta, \hat{E}_g^\beta) | \beta = 0, 1, 2, \cdots, m\}$. This process is defined as a *configuration decomposition* for $G_i = (V_i, E_i)$ and $G_g = (V_g, E_g)$ with respect to the root module pair τ_i and τ_g written as $\mathcal{CD}(G_i, \tau_i, G_g, \tau_g)$. This configuration decomposition can be finished in time $\mathcal{O}(|V_i|^2)$ or $\mathcal{O}(|E_i|^2)$ and, in reality, a SMORES-EP module has only 4 connectors so the time for a large number of modules should be much smaller than the worst case [22].

B. Module Mapping

Applying configuration decomposition for the initial and goal configurations $G_i = (V_i, E_i)$ and $G_g = (V_g, E_g)$, modules involved in $\text{MCS}(\tau_i, \tau_g)$ are mapped under $f : \bar{V}_i \to \bar{V}_g$ where $\bar{V}_i \subseteq V_i$ and $\bar{V}_g \subseteq V_g$, so $\hat{G}_i = (\hat{V}_i, \hat{E}_i)$ and $\hat{G}_g = (\hat{V}_g, \hat{E}_g)$ can be generated respectively, both of which are composed of multiple subconfigurations. The connection between $\bar{G}_i = (\bar{V}_i, \bar{E}_i)$ and subconfiguration $\hat{G}_i^\alpha = (\hat{V}_i^\alpha, \hat{E}_i^\alpha)$ in $\hat{G}_i = (\hat{V}_i, \hat{E}_i)$ for $\alpha = 0, 1, \cdots, n$ is denoted as $\text{connect}(u, \tau_i^\alpha)$ where $u \in \bar{V}_i$ and $\tau_i^\alpha \in \hat{V}_i^\alpha$. The vertex τ_i^α is called the *subconfiguration root* for $G_i = (V_i, E_i)$ and there are n subconfiguration roots in total. Similarly there are m subconfiguration roots for $G_g = (V_g, E_g)$ denoted as τ_g^β where $\beta = 0, 1, \cdots, m$.

We can then replace $\bar{G}_i = (\bar{V}_i, \bar{E}_i)$ with a virtual module $\mathcal{M}$ and replace $\text{connect}(u, \tau_i^\alpha)$ with a virtual connection $\text{connect}(\mathcal{M}, \tau_i^\alpha)$ defined as

$$\begin{aligned}\text{connect}(\mathcal{M}, v) = &\{\text{Face} : Null, \text{Face2Con} : Null, \\ &\text{Orientation} : Null\}\end{aligned} \tag{3}$$

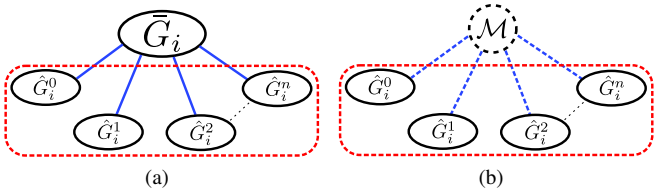


Fig. 7. Replace $\bar{G}_i = (\bar{V}_i, \bar{E}_i)$ with virtual module $\mathcal{M}$ and replace connections between $\bar{G}_i = (\bar{V}_i, \bar{E}_i)$ and every $\hat{G}_i^\alpha = (\hat{V}_i^\alpha, \hat{E}_i^\alpha)$ with virtual connections.

All virtual connections are equivalent. The new corresponding modular robot configuration with a virtual module and some virtual connections is written as $G_i' = (V_i', E_i')$ where $V_i' = V_i \setminus \bar{V}_i \cup \{\mathcal{M}\}$ as shown in Fig. 7. We call this operation the *Virtual Module Operation*.

These procedures can be applied to the goal configuration $G_g = (V_g, E_g)$ and the corresponding $G_g' = (V_g', E_g')$ is generated. Applying configuration decomposition on $G_i' = (V_i', E_i')$ and $G_g' = (V_g', E_g')$ with respect to virtual module $\mathcal{M}_i$ and $\mathcal{M}_g$, $\mathrm{MCS}(\mathcal{M}_i, \mathcal{M}_g)$ under mapping $f : \bar{V}_i' \to \bar{V}_g'$ where $\bar{V}_i' \subseteq V_i'$ and $\bar{V}_g' \subseteq V_g'$ can be computed. The solution to $\mathrm{MCS}(\mathcal{M}_i, \mathcal{M}_g)$ may not be unique. In addition to these two virtual modules, each vertex $u \in \bar{V}_i'$ is mapped to a unique module $v \in \bar{V}_g'$. After configuration decomposition $\mathcal{CD}(G_i', \mathcal{M}_i, G_g', \mathcal{M}_g)$, $\hat{G}_i' = (\hat{V}_i', \hat{E}_i')$ and $\hat{G}_g' = (\hat{V}_g', \hat{E}_g')$ are generated respectively. We can repeat the virtual module operation for $\hat{G}_i' = (\hat{V}_i', \hat{E}_i')$ and $\hat{G}_g' = (\hat{V}_g', \hat{E}_g')$ and two new modular robot configurations with virtual modules and virtual connections are generated.

Mapping is completed by repeating this process until every module in V_i has been mapped with a unique module in V_g. If we assume configuration decomposition is applied N times, then N mappings $f_1, f_2, \cdots, f_N$ are computed in order. This mapping process maintains the vertex height between configurations as much as possible and also keeps most of the common topology connections so that fewer reconfiguration actions are needed. For the worst case, the mapping process has to do configuration decomposition $\lceil |V_i|/2 \rceil + 1$ times (at least two modules can be mapped after each configuration decomposition except for the first and the last configuration decomposition and $\lceil x \rceil$ maps x to the least integer greater than or equal to x), so the time complexity is $\mathcal{O}(|V_i|^3)$. Again, for a large number of modules, in reality the mapping process should be much faster than the worst case.

C. Reconfiguration Actions

Once the mapping process is done, corresponding reconfiguration actions can be determined. Assume there are N mappings $f_t : {}^tV_i \to {}^tV_g$, $t = 1, 2, \cdots, N$ computed in order, then a mapping $f : V_i \to V_g$ that maps all modules from the initial configuration $G_i = (V_i, E_i)$ to the goal configuration $G_g = (V_g, E_g)$ can be obtained by the combination of these mappings while excluding virtual module mapping $(\mathcal{M}_i \to \mathcal{M}_g)$. This mapping is one-to-one and onto and the inverse of the mapping is $f^{-1} : V_g \to V_i$. Thus the reconfiguration actions can be computed by iterating modules in $G_i = (V_i, E_i)$ from leaves to the root.

For a modular robot configuration $G = (V, E)$ rooted at τ, for any vertex $v \in V$ with depth $d(v) > 0$, we denote its parent connected via its connector c as $\tilde{v}^c$ and the mating connector of $\tilde{v}^c$ as $\tilde{c}$. Given the mapping $f : V_i \to V_g$, each $v_i \in V_i$ is mapped to a unique $v_g \in V_g$, and $\tilde{v}_i^{c_i}$ and $\tilde{v}_g^{c_g}$ are their parents respectively. Similarly, with the inverse mapping $f^{-1} : V_g \to V_i$, $\tilde{v}_g^{c_g} \in V_g$ is also mapped to a unique $v_i' \in V_i$. If module pair (v_i, v_g) and $(\tilde{v}_i^{c_i}, \tilde{v}_g^{c_g})$ are in any MCS during the module mapping process, then $\mathrm{connect}(v_i, \tilde{v}_i^{c_i}) \cong \mathrm{connect}(v_g, \tilde{v}_g^{c_g})$ and there is no need to reconfigure. Otherwise, the reconfiguration actions are undocking v_i from $\tilde{v}_i^{c_i}$ by removing $\mathrm{connect}(v_i, \tilde{v}_i^{c_i})$ and docking v_i with v_i' by constructing $\mathrm{connect}(v_i, v_i')$

Once all modules except subconfiguration roots for $G_i = (V_i, E_i)$ and modules in $\mathrm{MCS}(\tau_i, \tau_g)$ are visited, $\hat{G}_i = (\hat{V}_i, \hat{E}_i)$ has reconfigured into $\hat{G}_g = (\hat{V}_g, \hat{E}_g)$ by executing reconfiguration actions from leaves to subconfiguration roots. This enables us to pick one solution to $\mathrm{MCS}(\mathcal{M}_i, \mathcal{M}_g)$ in the module mapping process freely since the module must be free to maneuver when a reconfiguration action is applied. We then execute the *Matching and Mapping* algorithm in [22] to check if the new configuration is isomorphic to the goal configuration. If not, we continue iterating unvisited modules and executing reconfiguration actions. This process can be done in time $\mathcal{O}(|V_i|)$.

D. Hardware Execution

To implement the reconfiguration plan described above with SMORES-EP, modules must undock from their initial positions in the current configuration, safely navigate them to their final positions in the goal configuration, and then dock to the appropriate modules.

The environment in which the modules will reconfigure can be described with a discrete representation. The graph representation of the environment, and the reconfiguration plan, can then be used to sequentially generate trajectories that safely navigate the modules to their new reconfigured positions in the goal configuration. These trajectories can be generated through graph search techniques such as A^* [28].

When generating the trajectory for each module, the modules not involved in the current reconfiguration action will be represented as static obstacles in the discrete environment. It may be the case that the current reconfiguration action requires the motion of other modules to create appropriate free-space. Given the full discrete environment and system state knowledge, a state machine can be used to identify when additional free-space is required, and initiate the motion of the occluding modules to enable the current reconfiguration action.

Since locomotion on the ground is achieved by rotating the LEFT and RIGHT Faces, aligning the orientation of that face with a stationary mating face could be problematic since their orientation is coupled to translation. When a mobile module is docking to a stationary one, if the face on the stationary module is a LEFT, RIGHT or TOP face, the stationary face can rotate to align the orientation appropriately. There are

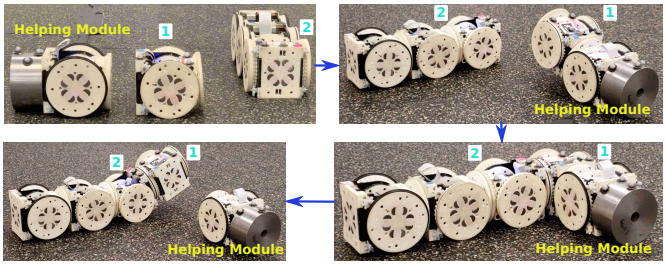
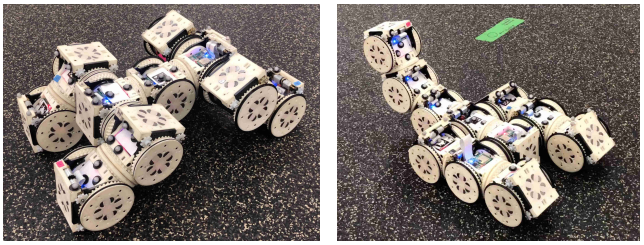


Fig. 8. A helping module docks with Module 1 BOTTOM Face and lifts it up so that LEFT Face of Module 1 can be aligned with BOTTOM Face of Module 2, then carry Module 1 to the location to finish the docking action.



(a) (b)

Fig. 9. Reconfigure a walker configuration (a) into a mobile vehicle with an arm configuration (b) with eleven SMORES-EP modules involved.

two special docking action cases in which this cannot be done: 1. Dock either a LEFT Face or a RIGHT Face with a mating BOTTOM Face and 2. Dock BOTTOM Face with a mating BOTTOM Face with Orientation attribute being 1. In these cases, the BOTTOM Face cannot rotate as the SMORES-EP modules have fixed BOTTOM faces. For these two cases, we utilize a helping module. A helping module is also a SMORES-EP module with some payload attached to its BOTTOM Face. The helping module can dock with and lift the mobile module so that the face is no longer coupled with the ground and can be orientated appropriately. A demo of this behavior is shown in Fig. 8.

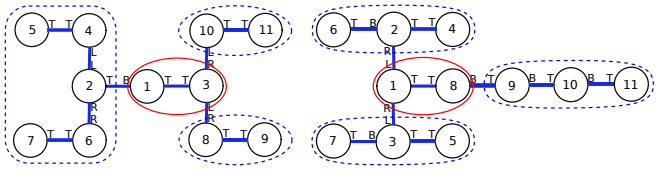
VI. Experiments

A ROS package for reconfiguration planning of SMORES-EP modules has been developed, which includes the reconfiguration planning algorithm, high-level mobile robot controller and low-level SMORES-EP module controller. This package is used to demonstrate three reconfiguration tasks.

1) Task 1 — Walker → Mobile Manipulator: Reconfigure a cluster of SMORES-EP modules from a walker (Fig. 9a) into a mobile vehicle with an arm (Fig. 9b) with eleven modules.

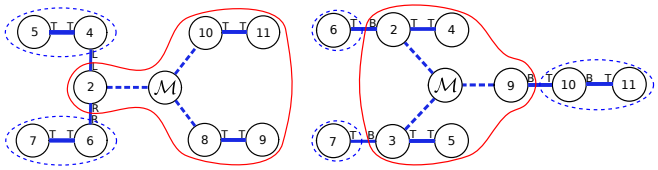
The initial and goal graph representations are shown in Fig. 10 where τ_i is Module 1 and τ_g is Module 1 respectively. For $G_i = (V_i, E_i)$ and $G_g = (V_g, E_g)$, $\mathrm{MCS}(\tau_i, \tau_g)$ only contains two modules under mapping $1 \to 1$ and $3 \to 8$. The result of the virtual module operation is shown in Fig. 11. In addition to some equivalent virtual connections, there are two common connections in $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$.

The rest of module mapping process is shown in Fig. 12 and Fig. 13. There are only virtual connections in $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$, each of which requires reconfiguration actions.



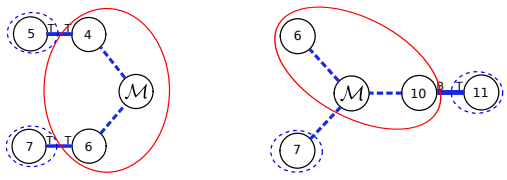
(a) (b)

Fig. 10. (a) Graph representation of walker configuration and (b) graph representation of mobile manipulator configuration. $\mathrm{MCS}(1,1)$ is encircled by "—" under mapping $1 \to 1$ and $3 \to 8$. After removing $\mathrm{MCS}(1,1)$, there are three unconnected subgraphs in both the current initial configuration and goal configuration which are encircled by "- -".



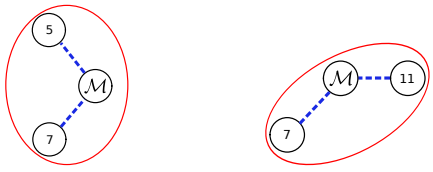
(a) (b)

Fig. 11. (a) New graph representation of walker configuration and (b) new graph representation of mobile manipulator configuration. $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$ is encircled by "—" under mapping $\mathcal{M} \to \mathcal{M}$, $9 \to 5$, $8 \to 3$, $2 \to 9$, $11 \to 4$ and $10 \to 2$. After removing $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$, there are two unconnected subgraphs in the current initial configuration and three unconnected subgraphs in the current goal configuration which are encircled by "- -".



(a) (b)

Fig. 12. $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$ is encircled by "—" under mapping $\mathcal{M} \to \mathcal{M}$, $4 \to 6$ and $6 \to 10$. After removing $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$, there are two unconnected subgraphs in both the current initial configuration and current goal configuration which are encircled by "- -"



(a) (b)

Fig. 13. $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$ is encircled by "—" under mapping $\mathcal{M} \to \mathcal{M}$, $5 \to 7$ and $7 \to 11$.

The final mapping is $f : V_i \to V_g$ is $1 \to 1$, $3 \to 8$, $9 \to 5$, $8 \to 3$, $2 \to 9$, $11 \to 4$, $10 \to 2$, $4 \to 6$, $6 \to 10$, $5 \to 7$ and $7 \to 11$ and the corresponding reconfiguration actions are shown in Table I.

The hardware execution of this plan is shown in Fig. 14. A VICON motion capture system is used to track the poses of modules. First, Module 6 and Module 7 have to move away so that Module 5 can move to dock with Module 8. Then Module 7 moves to dock with BOTTOM Face of Module 6, Module 4 undocks from Module 2 and moves to dock with Module 10. Finally Module 6 moves to dock with Module 2 followed

TABLE I
RECONFIGURATION ACTIONS FOR TASK 1

Action	ID	Face	ID	Face	Orientation
Undock	5	TOP Face	4	TOP Face	Null
Dock	5	TOP Face	8	BOTTOM Face	Null
Undock	7	TOP Face	6	TOP Face	Null
Dock	7	TOP Face	6	BOTTOM Face	Null
Undock	4	LEFT Face	2	LEFT Face	Null
Dock	4	TOP Face	10	BOTTOM Face	Null
Undock	6	RIGHT Face	2	RIGHT Face	Null
Dock	6	TOP Face	2	BOTTOM Face	Null

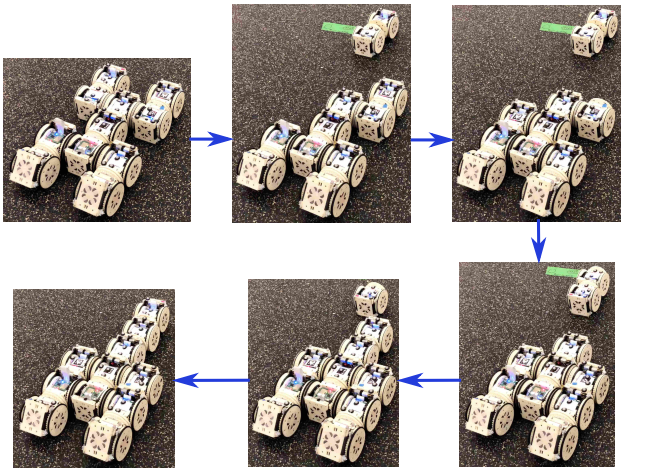


Fig. 14. SMORES-EP hardware reconfiguration from a walker to a mobile vehicle with an arm.

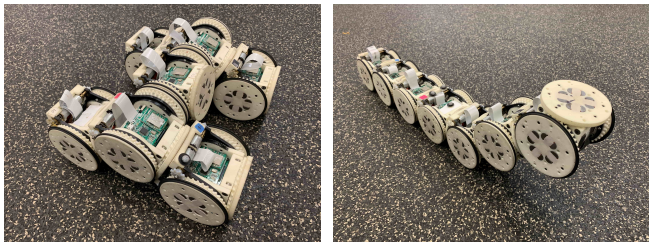
TABLE II
VERTEX HEIGHT OF MODULES IN $G_i(V_i, E_i)$ AND $G_g(V_g, E_g)$

v	1	2	3	4	5	6	7	8	9	10	11
$h(v)$ in G_i	3	2	2	1	0	1	0	1	0	1	0
$h(v)$ in G_g	3	2	4	0	0	1	0	1	0	1	0

by module 7. Now $\hat{G}_g = (\hat{V}_g, \hat{E}_g)$ is formed and we run the Matching and Mapping algorithm which shows that this configuration is isomorphic to $G_g = (V_g, E_g)$ and no further reconfiguration actions are needed. Table II shows how the vertex height of each module changes after the reconfiguration actions and, for those modules which need to execute actions, their heights have no change except for Module 4.

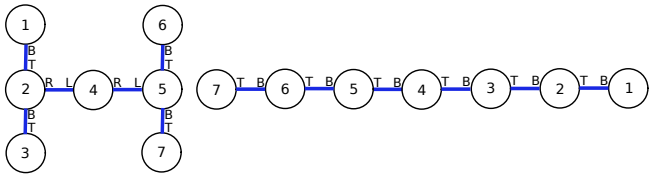
2) Task 2 — Driver → Snake: Reconfigure a cluster of SMORES-EP modules from a driver (Fig. 15a) into a snake (Fig. 15b) with seven modules.

The initial and goal graph representations are shown in Fig. 16 where τ_i is Module 4 and τ_g is Module 4 respectively. $\mathrm{MCS}(\tau_i, \tau_g)$ is empty. A virtual module $\mathcal{M}$ and a virtual connection $(\mathcal{M}, \tau_i)$ are added to G_i and a similar operation is applied to G_g. Then $\mathrm{MCS}(\mathcal{M}, \mathcal{M})$ is under mapping $\mathcal{M} \to \mathcal{M}$ and $4 \to 4$ which can be removed for further configuration decomposition. The final mapping between these two configurations is $f : V_i \to V_g$ is $1 \to 7$, $2 \to 5$, $3 \to 6$, $4 \to 4$, $5 \to 3$, $6 \to 2$ and $7 \to 1$ and the corresponding reconfiguration actions are shown in Table III.



(a) (b)

Fig. 15. Reconfigure a driver configuration (a) into a snake configuration (b) with seven SMORES-EP modules involved.



(a) (b)

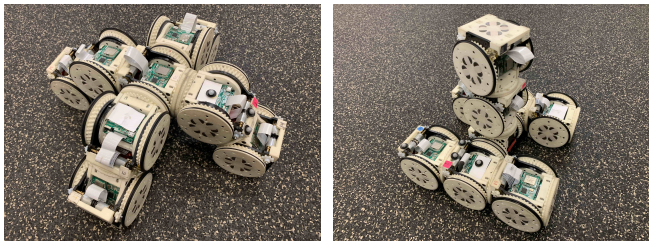
Fig. 16. (a) Graph Representation of Driver Configuration and (b) Graph Representation of Snake Configuration.

TABLE III
RECONFIGURATION ACTIONS FOR TASK 2

Action	ID	Face	ID	Face	Orientation
Undock	1	BOTTOM Face	2	TOP Face	Null
Dock	1	TOP Face	3	BOTTOM Face	Null
Undock	7	TOP Face	5	BOTTOM Face	Null
Dock	7	BOTTOM Face	6	TOP Face	Null
Undock	2	RIGHT Face	4	LEFT Face	Null
Dock	2	TOP Face	4	BOTTOM Face	Null
Undock	5	LEFT Face	4	RIGHT Face	Null
Dock	5	BOTTOM Face	4	TOP Face	Null

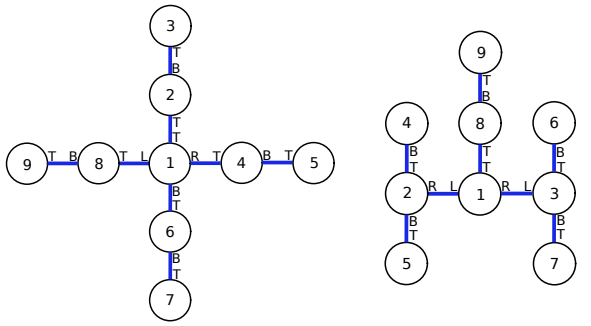
3) Task 3 — Omni-Driver → Mobile Observer: Reconfigure a cluster of SMORES-EP modules from a omni-driver (Fig. 17a) into a mobile observer (Fig. 17b) with nine modules.

The initial and goal graph representations are shown in Fig. 18 where τ_i is Module 1 and τ_g is Module 1 respectively. $\mathrm{MCS}(\tau_i, \tau_g)$ is under mapping $1 \to 1$, $2 \to 8$ and $3 \to 9$ which can be maintained during the reconfiguration process. The final mapping for other modules is $4 \to 2$, $5 \to 5$, $6 \to 4$, $7 \to 6$, $8 \to 3$ and $9 \to 7$, and the corresponding reconfiguration actions are shown in Table IV.



(a) (b)

Fig. 17. Reconfigure a omni-driver configuration (a) into a mobile observer configuration (b) with nine SMORES-EP modules involved.



(a) (b)

Fig. 18. (a) Graph Representation of Omni-Driver Configuration and (b) Graph Representation of Mobile Observer Configuration.

TABLE IV
RECONFIGURATION ACTIONS FOR TASK 3

Action	ID	Face	ID	Face	Orientation
Undock	7	TOP Face	6	BOTTOM Face	Null
Dock	7	BOTTOM Face	8	TOP Face	Null
Undock	8	TOP Face	1	LEFT Face	Null
Dock	8	LEFT Face	1	RIGHT Face	Null
Undock	4	TOP Face	1	RIGHT Face	Null
Dock	4	RIGHT Face	1	LEFT Face	Null
Undock	6	TOP Face	1	BOTTOM Face	Null
Dock	6	BOTTOM Face	4	TOP Face	Null

VII. CONCLUSION

In this paper, we present a new reconfiguration algorithm for modular robots. Graph representations of modular robot configurations are used and, based on our previous work, an efficient algorithm is developed to do configuration decomposition iteratively by adding virtual modules and virtual connections. Each module in the initial configuration is mapped to a module in the goal configurations with which reconfiguration actions can be computed. A helping module is designed to handle special reconfiguration actions. This algorithm is demonstrated with SMORES-EP hardware to show its effectiveness.

Future work will focus on shape morphing for complicated 3D structures so that a planar configuration can become an intermediate state between any two 3D configurations and this reconfiguration algorithm can be used to achieve chain-type or lattice-type reconfiguration motions.

REFERENCES

[1] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular Self-Reconfigurable Robot Systems: Grand Challenges of Robotics," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[2] K. Stoy, D. Brandt, and D. Christensen, *Self-Reconfigurable Robots*. Cambridge, MA: The MIT Press, 2010.

[3] S. Vassilvitskii, M. Yim, and J. Suh, "A Complete, Local and Parallel Reconfiguration Algorithm for Cube Style Modular Robots," in *Proc., 2002 IEEE Int. Conf. on Robotics and Automation*, 2002.

[4] Z. Bulter, K. Kotay, D. Rus, and K. Tomita, "Generic Decentralized Control for Lattice-Based Self-Reconfigurable Robots," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 919–937, 2004.

[5] F. R. and B. Z., "Million Module March: Scalable Locomotion for Large Self-Reconfiguring Robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 331–343, 2008.

[6] A. Naz, P. B., and B. J., "A Distributed Self-Reconfiguration Algorithm for Cylindrical Lattice-Based Modular Robots," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, 2016, pp. 254–263.

[7] M. Yim, D. Duff, and K. Roufas, "PolyBot: A Modular Reconfigurable Robot," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, April 2000, pp. 514–520 vol.1.

[8] S. Murata, K. Tomita, E. Yoshida, H. Kurokawa, and S. Kokaji, "Self-Reconfigurable Robot-Module Design and Simulation," in *Proc., 6th Int. Conf. on Intelligent Autonomous Systems*, 2000, pp. 911–917.

[9] B. Salemi, M. Moll, and W. Shen, "SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 3636–3641.

[10] M. Yim, P. White, M. Park, and J. Sastra, *Modular Self-Reconfigurable Robots*. New York, NY: Springer New York, 2009, pp. 5618–5631.

[11] N.-S. L. E., G. R., P. C. J. J., and K. P. K., "Modularity in Small Distributed Robots," *Proc.SPIE*, vol. 3839, pp. 3839 – 3839 – 10, 1999.

[12] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous Self-Assembly in a Swarm-Bot," in *Proc. of the 3rd Int. Symp. on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, K. Murase, K. Sekiyama, N. Kubota, T. Naniwa, and J. Sitte, Eds. Springer, Berlin, Germany, 2006, pp. 314–322.

[13] B. T. Kirby, B. Aksak, J. D. Campbell, J. F. Hoburg, T. C. Mowry, P. Pillai, and S. C. Goldstein, "A Modular Robotic System Using Magnetic Force Effectors," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 2787–2793.

[14] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A Low Cost Scalable Robot System for Collective Behaviors," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3293–3298.

[15] J. Davey, N. Kwok, and M. Yim, "Emulating Self-reconfigurable Robots — Design of the SMORES System," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4464–4469.

[16] A. Casal and M. Yim, "Self-Reconfiguration Planning for a Class of Modular Robots," vol. 3839, 1999, pp. 3839–3839–12.

[17] C. Nelson, "A Framework for Self-Reconfiguration Planning for Unit-Modular Robots," Ph.D. dissertation, Purdue University, West Lafayette, 2005.

[18] K. Payne, B. Salemi, P. Will, and W. Shen, "Sensor-Based Distributed Control for Chain-Typed Self-Reconfiguration," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, Sep. 2004, pp. 2074–2080 vol.2.

[19] M. Asadpour, M. Ashtiani, A. Sproewitz, and A. Ijspeert, "Graph Signature for Self-Reconfiguration Planning of Modules with Symmetry," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 5295–5300.

[20] F. Hou and W. Shen, "Distributed, Dynamics, and Autonomous Reconfiguration Planning for Chain-Type Self-Reconfigurable Robots," pp. 3135–3140, May 2008.

[21] ——, "Graph-Based Optimal Reconfiguration Planning for Self-Reconfigurable Robots," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 1047 – 1059, 2014.

[22] C. Liu and M. Yim, "Configuration Recognition with Distributed Information for Modular Robots," in *IFRR International Symposium on Robotics Research*, 2017.

[23] T. Tosun, J. Davey, C. Liu, and M. Yim, "Design and Characterization of the EP-Face Connector," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 45–51.

[24] G. Jing, T. Tosun, M. Yim, and H. Kress-Gazit, "An End-To-End System for Accomplishing Tasks with Modular Robots," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[25] T. Tosun, J. Daudelin, G. Jing, H. Kress-Gazit, M. Campbell, and M. Yim, "Perception-Informed Autonomous Environment Augmentation with Modular Robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6818–6824.

[26] M. Park, S. Chitta, A. Teichman, and M. Yim, "Automatic Configuration Recognition Methods in Modular Robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 403–421, 2008.

[27] G. McColm, "On the Structure of Random Unlabelled Acyclic Graphs," *Discrete Mathematics*, vol. 277, no. 1, pp. 147–170, 2004.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.