Proceedings of DETC'04
ASME 2004 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
September 28-October 2, 2004, Salt Lake City, Utah, USA

DETC2004-57488

AN ALGORITHM FOR EFFICIENT SELF-RECONFIGURATION OF CHAIN-TYPE UNIT-MODULAR ROBOTS

Carl A. Nelson
Purdue University
School of Mechanical Engineering
585 Purdue Mall
West Lafayette, Indiana 47907-2088
USA
carln@purdue.edu

Raymond J. Cipra
Purdue University
School of Mechanical Engineering
585 Purdue Mall
West Lafayette, Indiana 47907-2088
USA
cipra@ecn.purdue.edu

Keywords: modular robots, reconfigurable robots, reconfiguration algorithms, chain-type robots.

ABSTRACT

The problem of self-reconfiguration of modular robots is discussed, and an algorithm for efficient parallel self-reconfiguration is presented. While much of the previous work has been focused on the lattice-type modular robots, this paper addresses the self-reconfiguration of chain-type robots. Relatively little attention has heretofore been given to this sub-problem, and of the existing work, none incorporates the kinematic limitations of real-life robots into the reconfiguration algorithm itself. The method presented here is based on understanding a robot's physical "composition" using a graph-theoretic robot representation, and it sheds new light on self-reconfiguration of chain-type modular robots by incorporating elements of the robot kinematics as part of the criteria in choosing reconfiguration steps.

INTRODUCTION

Unit-modular robots are defined as robots composed of identical, standard component modules [1]. This type of robot is preferred for its fault tolerance, versatility to different tasks, and simplified kinematics and dynamics. Due to the uniformity of the modules, such a robot can be reconfigured into a large number of useful forms and topologies and assume a variety of locomotion modes through changing the module interconnections. A special class of modular robots are called self-reconfiguring, since they are able to achieve such reconfiguration without human aid or interaction.

Robotic reconfiguration can be thought of as an abstraction of certain phenomena occurring in natural systems. Two examples are blood clotting and chemical reactions. In the process of blood clotting, platelet cells come together to form interwoven chains, trapping the rest of the blood and preventing it from escaping. In chemical reactions, bonds are broken and formed between atoms, giving rise to different chemical species. In a similar way, a randomly arranged uniform set of robotic modules can rearrange to form chains or molecule-like structures.

The majority of the research in self-reconfiguring modular robots has been centered on the so-called lattice robots. These are robots whose modules can assume finite positions in a grid, either in planar or three-dimensional Cartesian space. A common example of this type of robotic module is a cube which can expand and form interconnections along each of its six cardinal directions. Locomotion and reconfiguration in lattice robots are achieved by motion of modules from one set of grid locations to another. Several reconfiguration algorithms for this type of robot have been proposed. Pamecha et al. [2] used the technique of simulated annealing to minimize a distance metric between configurations. Yoshida et al. [3] used a technique involving potential fields and weighted probabilities called stochastic relaxation to guide modules toward a goal configuration. Vassilvitskii et al. [4] presented a rule-based approach for planning and ordering reconfiguration steps. Rule-based algorithms for self-replication (division), locomotion, and recombination of lattice robots were given by Butler et al. [5]. Reconfiguration control algorithms based on "goal ordering" were explored by Yim et al. [6], and Bojinov et al. [7] investigated biologically inspired approaches to reconfiguration in which the final robot configuration is not known a priori. Locomotion of lattice-type robots through reconfiguration has also been explored [8], and a lattice-robot reconfiguration strategy involving "tunneling" has been presented by Nguyen et al. [9].

The present work focuses on chain-type modular robots having one rotational degree of freedom per module, two configurations of which are shown in Figure 1. The reconfiguration problem for chain-type robots is somewhat different from that of lattice-type robots. Chain-type modular robots are not defined by their position in a grid. Instead, due to serial and/or parallel connections of modules using continuous joints such as revolute or spherical joints, they exist in a continuous rather than a discrete space and can form loops, chains, or truss-type structures. However, their various configurations are discretely different from each other. By their nature, chain-type robots generally locomote through continuous joint motions and reconfigure through discrete "connect" and "disconnect" operations at the joints. Thus the reconfiguration problem, instead of being very discrete in nature, is a combination of a continuous kinematics problem and a discrete connection-state problem. For this reason, the previous research in lattice-type reconfiguration is not very useful in addressing the problem of chain-type reconfiguration.

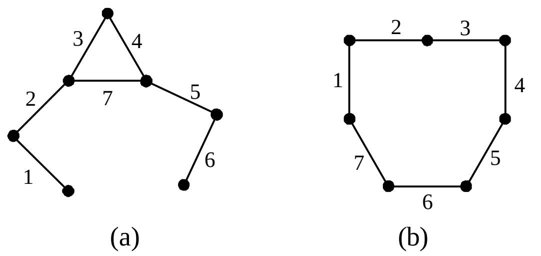


Figure 1. Two planar chain-type robot configurations: (a) walker, and (b) tank tread.

Casal and Yim [1] presented two rule-based reconfiguration algorithms, and Yim et al. [10] presented a third algorithm for self-reconfiguration in this type of robot. The first of these involves reducing the robot from its initial configuration to a simple "canonical form" and then building in complexity toward the goal shape. The second algorithm involves identifying substructures within the goal configuration (e.g. loops and open-ended chains) and then choosing reconfiguration steps so as to match the number and location of these substructures. The third algorithm is essentially an improved form of the first, where the canonical form is a tree structure (a graph with no closed cycles). These three algorithms, while adequate in solving the reconfiguration problem, have some important limitations. The first two are in fact rather inefficient by their nature. The third, because of a wise choice of canonical form, does better in this respect. None of them are parallel algorithms, which severely decreases the speed at which they can be executed.

Durna et al. [11] demonstrated the use of the Hungarian algorithm, popular in transport theory, for the self-reconfiguration problem. This algorithm, while popular and proven, is not parallel, and in some other ways is not well suited for the chain-robot reconfiguration problem. These same researchers also used the robot's eigenvalue and eigenspace information to steer a colony of robotic modules toward a goal configuration [12]. While this approach is distributed and parallel, the manner in which the control is distributed severely limits the number of choices of reconfiguration steps, hence rendering it ineffective for certain reconfiguration tasks.

The aim of this paper is to present a new strategy for self-reconfiguration of chain-type unit-modular robots. Its effectiveness and uniqueness stem from two characteristics of the strategy. First, the algorithm takes into account the limitations of the robot kinematics in its choice of reconfiguration steps. This is done by first using the robot's graph-theoretic representation to approximate the feasibility of potential reconfiguration steps, and then performing the kinematic calculations necessary to execute these steps. Second, the algorithm finds compatible groups of reconfiguration steps and then performs these steps in parallel, thus reducing the time required to complete the reconfiguration.

The following description of the strategy addresses several sub-problems. First, a set of rules for mathematical representation of robots is presented. This representation is compared and contrasted to existing representation techniques for closed-loop mechanisms. A technique for determining the mechanical composition of robots is presented, and the initial sub-problem of "matching" two dissimilar robotic configurations is investigated. Finally, the reconfiguration algorithm is detailed and examples given to demonstrate its use.

GRAPH REPRESENTATION OF MODULAR ROBOTS

Because a modular robot is a set of identical components connected by joints, it is well suited to representation using graph theory [11-12]. A graph consists of n vertices joined by m edges. It is said to be connected if there exists a path along the graph's edges which connects any arbitrary pair of vertices. If a single robotic module is represented by a vertex and its joint with another module is represented by an edge, the entire robot is represented by a connected graph.

The graph of a robot can also be represented in matrix form. The adjacency matrix A of a graph is a symmetric n×n matrix in which each of the n vertices (robotic link modules) is represented by its own row (and column). If vertex i is incident on vertex j by an edge, then entry $a_{ij} = 1$; otherwise, $a_{ij} = 0$. Another useful matrix representation of modular robots is the edge-vertex (joint-link) incidence matrix (IM). For a robot with m joints, each row of the m×n incidence matrix represents a joint between robotic modules, and the columns correspond to the link modules. In this way, $IM_{ij} = 1$ if joint i contains link j; otherwise $IM_{ij} = 0$.

For example, the adjacency and incidence matrices for the robot shown in Figure 1(a) are

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad IM = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (1)$$

and the matrices corresponding to Figure 1(b) are

$$A=\begin{bmatrix}0&1&0&0&0&0&1\\1&0&1&0&0&0&0\\0&1&0&1&0&0&0\\0&0&1&0&1&0&0\\0&0&0&1&0&1&0\\0&0&0&0&1&0&1\\1&0&0&0&0&1&0\end{bmatrix},\quad IM=\begin{bmatrix}1&1&0&0&0&0&0\\0&1&1&0&0&0&0\\0&0&1&1&0&0&0\\0&0&0&1&1&0&0\\0&0&0&0&1&1&0\\0&0&0&0&0&1&1\\1&0&0&0&0&0&1\end{bmatrix} \quad (2)$$

Both the adjacency matrix and incidence matrix representations are useful for describing modular robots in mathematical form.

GRAPH REPRESENTATION OF CLOSED-LOOP MECHANISMS

Previous investigators [13-14] have used graph theory to represent closed-loop mechanisms in order to systematically analyze their motion and governing dynamic equations. It is important to note the differences between these methods and the technique of graph representation of modular robots. For closed-loop mechanisms, joints are modeled as graph edges, and links as vertices. Based on the cycles of the resulting graph (closed paths beginning and ending at the same vertex), the mobility and motion properties of the mechanism can be determined. The information from the graph can ultimately be used to generate kinematic and dynamic equations describing the mechanism. However, modular robots can form closed-loop mechanisms, open chains, or combinations of the two. For this reason, the graph "bookkeeping" is slightly different.

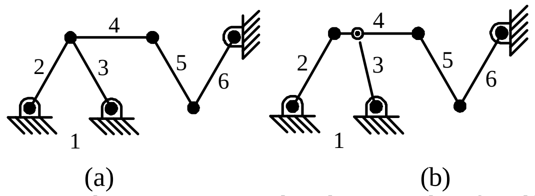


Figure 2. Single-DOF mechanism made of uniform link modules: (a) with and (b) without special joint-location geometry.

Consider the example of Figure 2, in which the mechanism has mobility 1. The equation for mobility of planar mechanisms with single-DOF joints is

$$M = 3(n - 1) - 2j \quad (3)$$

where n is the number of links and j is the number of joints. For this equation to give the correct result for the mechanism in Figure 2(a), the (2,3,4) joint must only count as two joints. This is because the second joint need not occur at the same location as the first, i.e. the (3,4) joint could occur in the middle of link 4, as shown in Figure 2(b). However, if the same mechanism were constructed of uniform robotic modules, the robot's adjacency matrix would show a joint between links 2 and 3, a joint between links 2 and 4, and a joint between links 3 and 4, for a total of three joints at the (2,3,4) connection. This is because the joints necessarily occur at the link endpoints, and because the possibility for open-chain robots exists. This "extra" piece of information contained in the robot's adjacency matrix is essential for extracting geometric information about the robot, i.e. which modules are connected at which endpoints. From this point, the physical embodiment of the robot can be visualized or simulated, since the use of uniform-length modules gives the rest of the necessary geometric information.

DETERMINING THE MECHANICAL COMPOSITION OF THE ROBOT CONFIGURATION

As mentioned above, a modular robot can contain truss-like substructures, open-ended chains, and closed-loop mechanisms. The occurrence of loops and chains has been recognized and capitalized on to some extent in the self-reconfiguration studies of Casal and Yim [1] and Yim et al. [10]. For the purposes of robotic reconfiguration, it is important to know which robot modules belong to each of these three groups. Most importantly, one must discern between substructures and mobile portions of the robot.

The easiest way to identify structural parts of the robot is to search for triangles, since the most basic immobile subset of uniform robotic modules is an equilateral triangle. This search can be done using the adjacency matrix and/or the joint-link incidence matrix. Once a triangle is identified, it represents a substructure. By searching for pairs of links which are each incident on the substructure at different locations and which are also incident on each other, the recognized bounds of the substructure grow until no more links can be added to it. This process is repeated until all of the robot's substructure parts are identified. Because this search technique begins at the center of each substructure and moves outward, the modules forming the perimeter of each substructure are also identified.

THE DIFFERENCE MATRIX AND SUBGRAPH MATCHING

Given two dissimilar robot configurations, it is useful to have a way of measuring or quantifying to what extent and in what ways the configurations differ. This information can serve as a guide to the reconfiguration process. For two different configurations of n robotic modules, an n×n difference matrix D can be defined as

$$D = A_{\mathrm{final}} - A_{\mathrm{initial}} \quad (4)$$

in which the nonzero entries in D represent dissimilarities between the initial and final graph configurations. For example, given the two robots in Figure 1,

$$D=\begin{bmatrix}0&1&0&0&0&0&1\\1&0&1&0&0&0&0\\0&1&0&1&0&0&0\\0&0&1&0&1&0&0\\0&0&0&1&0&1&0\\0&0&0&0&1&0&1\\1&0&0&0&0&1&0\end{bmatrix}-\begin{bmatrix}0&1&0&0&0&0&0\\1&0&1&0&0&0&1\\0&1&0&1&0&0&1\\0&0&1&0&1&0&1\\0&0&0&1&0&1&1\\0&0&0&0&1&0&0\\0&1&1&1&1&0&0\end{bmatrix} \quad (5)$$

$$=\begin{bmatrix}0&0&0&0&0&0&1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&1\\1&-1&-1&-1&-1&1&0\end{bmatrix}$$

This definition of D depends on proper ordering of the modules. The difference matrix is only useful for guiding reconfiguration when the modules of the initial and final configurations are ordered in such a way that the norm of the difference matrix is minimum. In graph-theoretic terms, this is called finding the maximal common subgraph of the graphs corresponding to A_{final} and $A_{initial}$.

Sometimes the robotic modules are ordered "nicely" such that the norm of D is relatively small, as is the case in Figure 1 and Eqs. (1-2, 5). For arbitrary module ordering in the adjacency matrices, the norm of D is not necessarily a minimum, and it does not accurately describe the "difference" between two robotic configurations. This would result in wasted motion if it were used as a guide for reconfiguration steps. In the case of arbitrary module ordering, this problem of finding the difference matrix can be formulated as an optimization problem of the form [11]

$$\min \| P A_{final} P^T - A_{initial} \| \qquad (6)$$

where P is an n×n permutation matrix which effectively switches the order of the rows and columns of the final configuration's adjacency matrix, hence renumbering the links in the final configuration. Algorithms exist to perform this matching such that $\|D\|$ is minimum. For labeled graphs, such as those used to represent chemical reactions, relatively efficient algorithms can be used. For general graphs, however, such as those which represent uniform robotic modules, the problem is slightly more difficult. Although convergence to the solution is guaranteed, the problem is NP-complete; in fact, its complexity is O(n!). In order to make it more manageable, McGregor [15] suggests using a depth-first tree search of partial matchings with branch elimination based on comparison of the partial solutions against the best-to-date full solution candidate. Additionally, an initial guess at the module ordering can help accelerate convergence of the algorithm. This is the approach which is adopted in step 1 of the algorithm to be described below.

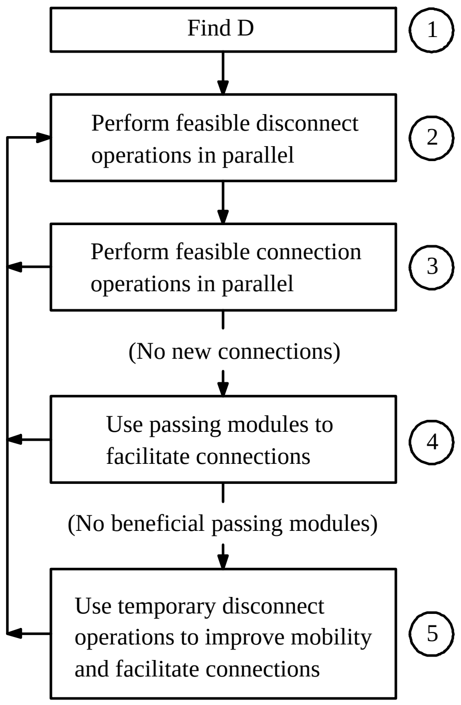
THE SELF-RECONFIGURATION ALGORITHM

Determination of the difference matrix D is the first step in the self-reconfiguration process. In terms of modules and joints, $d_{ij} = -1$ implies that modules i and j need to be disconnected during the process of self-reconfiguration, and $d_{ij} = 1$ implies that modules i and j need to be connected with a new joint during reconfiguration. To find the difference matrix, the procedure suggested by McGregor [15] is used, as described above.

Although in theory the reconfiguration is as simple as performing the set of disconnect and connect operations as given in the D matrix, in practice these operations cannot necessarily be performed at the same time or at all. The possibility of performing a disconnect operation is reliant on the overall connectivity of the robot's graph, and the possibility of making a new connection is reliant on factors such as robot workspace and environmental or internal obstacles. While the kinematics and path planning for robotic links is not specifically addressed in this paper, the goal of the algorithm is to order these robotic motions in such a way as to minimize the time and energy required to complete the reconfiguration. The following steps are proposed for self-reconfiguration while taking into account such kinematic limitations.

1. Find the difference matrix D using Eq. (4).
2. Judge the feasibility of the disconnect operations indicated in D. Perform all feasible disconnect operations in parallel, and update D.
3. Judge the feasibility of the connection operations indicated in D. Perform all feasible connection operations in parallel, and update D. Return to step 2 and repeat until there are no feasible connections to be made in step 3.
4. Remaining connections cannot be made directly. Use intermediate mobile links to "pass" modules to goal connection points. Update D and return to step 2, or continue to step 5 if no beneficial passing module is found.
5. If necessary, disconnect joints which are not marked for disconnecting in D; this improves the workspace of mobile links and allows more efficient "passing" in step 4. Update D and return to step 2.

The reconfiguration is finished when D contains all zero entries.



Find D
1
Perform feasible disconnect operations in parallel
2
Perform feasible connection operations in parallel
3
(No new connections)
Use passing modules to facilitate connections
4
(No beneficial passing modules)
Use temporary disconnect operations to improve mobility and facilitate connections
5

Figure 3. Flowchart for the reconfiguration algorithm.

A flowchart for these steps is shown in Figure 3, and their details are illustrated in a series of examples.

EXAMPLE 1: WALKER TO TANK TREAD

The robots shown in Figure 1 are used as a simple case study to illustrate steps (1-3) of the algorithm. This example involves no module "passing" and is the simplest scenario encountered with the above algorithm.

(1) The entries of D serve as a guide for the rest of the reconfiguration steps. From Eq. (5), the difference matrix is

$$D=\begin{bmatrix}0&0&0&0&0&0&1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&1\\1&-1&-1&-1&-1&1&0\end{bmatrix} \quad (7)$$

(2) A feasibility judgment for disconnect operations could include factors such as maintaining connectivity of the robot and staying within force limits of the joint actuators (for static stability). Here we consider only robot connectivity. Feasible sets of disconnect operations to be performed on the walker of Figure 1(a), which are indicated in D, are {(2,7),(3,7)} or {(4,7),(5,7)}. Performing all of these disconnect operations at once is not allowed, since it can be seen from Eq. (1) that doing so would result in an all-zero row/column, indicating a disconnected robot.

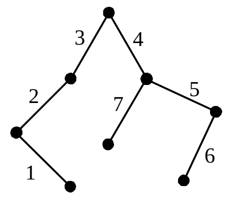


Figure 4. Disconnecting joints of the walker robot.

Arbitrarily we choose the former as shown in Figure 4. The new difference matrix is

$$D=\begin{bmatrix}0&0&0&0&0&0&1\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1\\0&0&0&0&0&0&1\\1&0&0&-1&-1&1&0\end{bmatrix} \quad (8)$$

(3) The next step involves performing sets of connection operations after first judging their feasibility. Here feasibility is determined using an approximation of the robot kinematics, taking into account possible internal interference in the robot. To do this, for each 2-module connection indicated in the D matrix, find a path through the robot's graph which connects the free ends of the two modules and passes only through mobile links and substructure perimeters, not through the center of substructures. For all the modules included in this path, subtract the number of substructure modules from the number of mobile modules. If the resulting quantity is positive, the connection is considered feasible. This represents an approximation of mobile links' capability to reach or wrap around structural parts of the robot in order to connect the desired modules. Note that modules “buried” inside a substructure cannot connect feasibly to other modules, since a path between the two modules does not exist which satisfies the constraints mentioned above. The “quality” of each potential connection is measured by the magnitude of the computed quantity based on the path through the robot (the larger that quantity the better). We call this the “connection quality index,” or q_c. For the disconnected walker of Figure 4, the path from link 1 to link 7 is (1,2,3,4), and the path from link 6 to link 7 is (6,5,7). (Note that since a connection between links 1 and 6 is not indicated in D, the path starting at link 1 and the path starting at link 6 end at opposite ends of link 7.) Since the entire robot is composed of mobile chains, the quality indices of these paths are $q_c = 4$ and $q_c = 3$ respectively, the former having higher “quality.” Now for the “best quality” potential connection, compute the robot kinematics (joint position solution) necessary for the connection and make a *virtual* connection; i.e. perform all future calculations as if this connection was actually made, but do not physically make the connection yet. This “virtual connection” step is repeated until there are no more feasible (virtual) direct connections to be made. This ensures that the maximum number of direct connections can be made at once. Now in parallel, perform this set of connections. This should be easy, since the joint position information is retained from the calculations performed during this step. For the example at hand, there are no substructures in the robot, and all the connections indicated in D are feasible. The result of making the (1,7) and (6,7) connections is shown in Figure 5. The updated difference matrix is

$$D=\begin{bmatrix}0&0&0&-1&-1&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\-1&0&0&0&0&0&-1\\-1&0&0&0&0&0&-1\\0&0&0&0&0&0&0\\0&0&0&-1&-1&0&0\end{bmatrix} \quad (9)$$

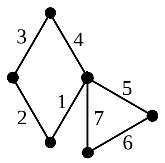


Figure 5. Feasible connections performed on the modified walker robot.

Note that connecting link 1 to links 4 and 5 is acceptable temporarily because link 7 is marked for disconnecting from these two links, so the (1,7) joint will separate from (4,5) as a single step.

(2) All the remaining disconnect operations are feasible, and the result is the goal configuration as shown in Figure 6.

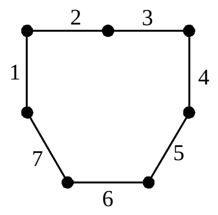


Figure 6. Tank-tread robot resulting from reconfiguration.

EXAMPLE 2: "PASSING" ROBOTIC MODULES

As shown in Figure 7, the task is to reconfigure the 9-link robot such that link 1 moves to the other side of the robot and connects to the (7,8,9) joint.

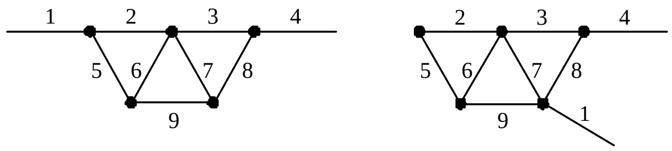


Figure 7. Initial and final configurations of the robot in Example 2.

(1) The adjacency matrices and difference matrix are

$$A_{initial} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (10)$$

$$A_{final} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (11)$$

$$D = \begin{bmatrix} 0 & -1 & 0 & 0 & -1 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (12)$$

(2) There are no feasible disconnect operations indicated in D since disconnecting the (1,2) joint or (1,5) joint still leaves links 2 and 5 connected, and disconnecting both the (1,2) and (1,5) joints results in a disconnected graph.

(3) The difference matrix indicates a connection to be made involving link 1 and joint (7,8,9). The shortest path from the end of link 1 to the (7,8,9) joint passes through one mobile link (1) and two links belonging to the robot's structural portion (5 and 9), giving a connection quality of $q_c = -1$. As shown in Figure 8, this does not result in a possible connection of link 1 with joint (7,8,9).

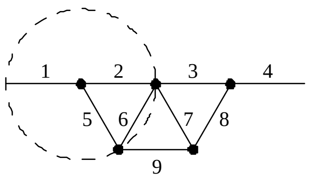


Figure 8. Check of the workspace of link 1.

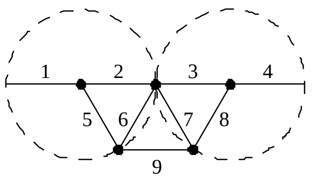


Figure 9. Check of mobile-link workspace overlap.

(4) The remaining connections cannot be made directly, so intermediate or "passing" modules must be used. To do this, use the path through the robot which was found in step 3. Note all the joints along the path where mobile portions of the robot attach to substructure perimeters. The "most mobile" of these mobile portions is used as a passing module. The number of robotic modules in a mobile portion can be used as an approximate measure of the quality of its mobility, so the mobile portion with the most modules is chosen as the passing module. The link module to be "passed" is connected to the link module on the mobile portion having the largest workspace (typically the end of a chain or middle link of a closed loop) and then connected to the goal link as indicated in the difference matrix. Multiple passing steps may be necessary. Since the path through the robot found in step 3 does not pass through any mobile sub-portions except link 1 itself, it is necessary to try a different path between link 1 and joint (7,8,9). (Link 1 could "pass itself" via the (5,6,9) joint, but this is impractical.) The other possible path includes links 2, 3, and 8, and passes by link 4, which is a mobile chain off of the substructure. Link 4 is chosen as a passing module. As shown in Figure 9, the workspaces of links 1 and 4 overlap, and link 1 can be passed (or connected temporarily) to link 4, as indicated in Figure 10(a). The path from link 1 to link 4 is (1,2,3,4) with a quality of $q_c = 0$ (barely feasible).

(2) Disconnect link 1 from links 5 and 2 as shown in Figure 10(b).

(3) Rotating the (1,4) sub-chain, connect link 1 to the joint at links (7,8,9).

(2) Disconnect links 1 and 4 as shown in Figure 10(c).

These final steps complete the reconfiguration.

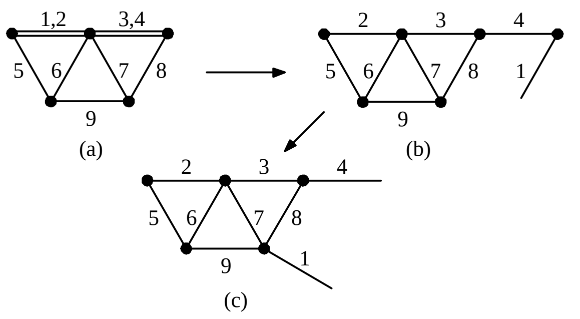


Figure 10. Final reconfiguration steps in Example 2.

EXAMPLE 3: IMPROVING PASSING MODULE MOBILITY THROUGH DISCONNECTING

As shown in Figure 11, the reconfiguration task is similar to that in Example 2, except that the robot's span is longer, leaving a larger distance for link 1 to traverse.

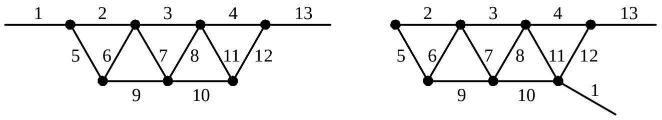


Figure 11. Initial and final robot configurations for Example 3.

(1) The difference matrix is similar to that in Example 2:

$$D = \begin{bmatrix} 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

(2) As in Example 2, there are no feasible disconnect operations indicated in D.

(3) The shortest path from the end of link 1 to the (10,11,12) joint passes through one mobile link (1) and three links belonging to the robot's structural portion (5,9,10), giving a connection quality of q_c = -2. As shown in Figure 12, this does not result in a possible connection with joint (10,11,12).

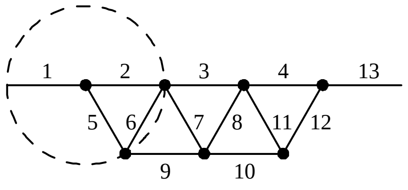


Figure 12. Check of workspace of link 1.

(4) The only chains attached to the substructure are link 13 and link 1 itself. However, as shown in Figure 13, the workspaces of links 1 and 13 are not close to overlapping. So it is advantageous to perform a temporary disconnect in order to increase the workspace of link 1.

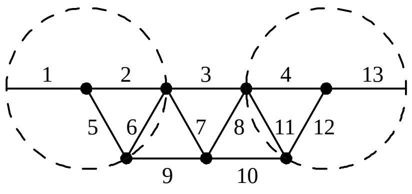


Figure 13. Check of mobile-link workspace overlap.

(5) In some cases, it may be necessary to temporarily disconnect some joints (which are not marked for disconnecting in the difference matrix D) in order to improve mobility (or workspace area) and facilitate passing of modules. This occurs in particular when multiple passing steps were found in step 4. If this point is reached, a judgment must be made as to which feasible disconnect would most improve the workspace and mobility properties and thus decrease or eliminate the passing steps (step 4). After performing this most advantageous disconnect operation, return to step 2. In this case, since link 5 is closer to the (10,11,12) joint than link 2, separating links 1 and 5 from link 2 is deemed most advantageous, and link 5 is used as an extension of link 1.

(2) Disconnect link 2 from links 5 and 1 as shown in Figure 14.

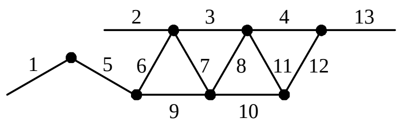


Figure 14. Temporary disconnect operation uses link 5 as a passing module.

(3) With these new disconnect operations performed, the path from link 1 to the (10,11,12) joint contains 2 mobile links and 2 structural links, giving a connection quality of q_c = 0. Since the two mobile links (1 and 5) form a simple chain, this indicates a "barely feasible" connection. Make this connection as shown in Figure 15(a).

(2) Disconnect links 1 and 5 as shown in Figure 15(b).

(3) Due to the temporary disconnect of link 5, there remains a connection to be made between links 2 and 5. The path between the two link endpoints contains 1 structural link and two mobile links, so the connection can be made as shown in Figure 15(c).

These final steps complete the reconfiguration.

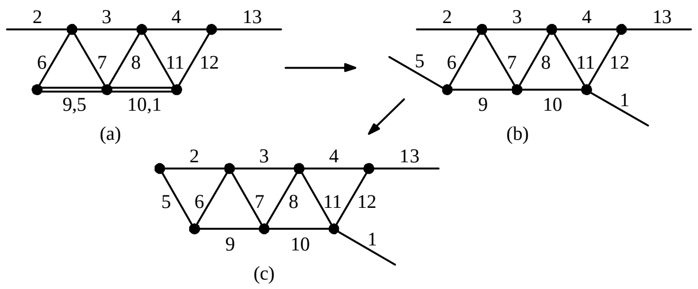


Figure 15. Final reconfiguration steps in Example 3.

EXAMPLE 4: DEMONSTRATION OF PARALLEL RECONFIGURATION STEPS

Two arbitrarily labeled 16-link robots are shown in Figure 16. Starting with an initial relabeling guess, a subgraph-matching algorithm was used to improve on that guess, resulting in the relabeled robots shown in Figure 17.

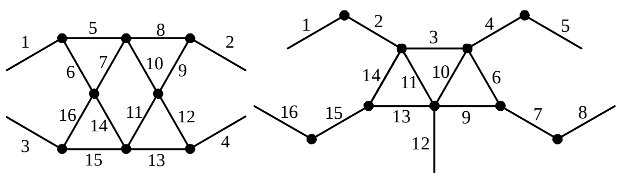


Figure 16. Initial and final robot configurations for Example 4.

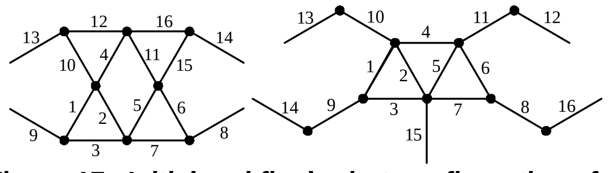


Figure 17. Initial and final robot configurations for Example 4 after relabeling.

(1) The norm of the difference matrix using this labeling scheme is 18. The disconnection entries in the lower triangular half of D are at {(12,10), (16,11), (15,11), (16,15), (16,14), (16,12), (16,4), (15,14), (15,6), (13,12), (12,4)}, and the connection entries in the lower triangular half of D are at {(16,8), (15,2), (15,3), (15,7), (14,9), (6,4), (5,4)}.

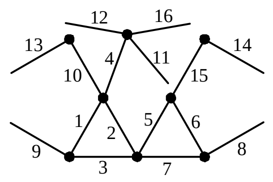


Figure 18. Parallel disconnect operations.

(2) As shown in Figure 18, link 12 can be disconnected from links 10 and 13, link 16 from links 15 and 14, and link 11 from links 5, 6, and 15, all in parallel. Note that the temporary (11,5) and (11,6) disconnects are not indicated in D. However, they are necessary in order to perform the (15,11) disconnect, which is chosen in order to preserve the (4,11) chain since the (4,5,6,11) joint occurs in the final robotic configuration. Although this type of logical decision-making is not discussed explicitly as part of step (2) of the algorithm, it is an example of how step (2) of the algorithm could potentially be refined.

(3) While link 16 can feasibly reach link 8, there is potential interference from the other links, especially 14 and 15. Also, connecting link 4 to links 5 and 6 results in several unwanted connections. So these direct connections are not made at this time.

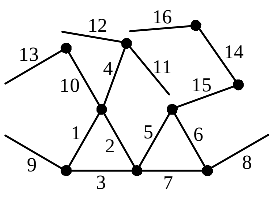


Figure 19. Using a passing module to circumvent internal interference.

(4) As shown in Figure 19, the (14,15) chain is used as a passing module to circumvent this internal interference problem and join links 16 and 8.

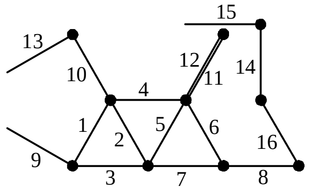


Figure 20. Parallel connections at (12,11) and (4,5,6,11,12).

(2) As depicted in Figure 20, the (15,14,16,8) chain is disconnected from (5,6).

(3) The (4,11,12) joint is also merged with the (5,6) joint after this last disconnect operation, as shown in Figure 20. Links 11 and 12 are also joined in order to move link 12 to the end of the chain.

(2) Link 12 is unfolded to form the (11,12) chain.

(4) Links 15 and 14 are passed to their final connection points using the (8,16) chain as a passing module, as shown in Figure 21.

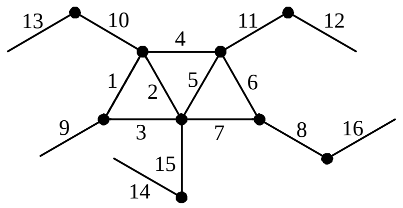


Figure 21. Passing module used for transport of links 14 and 15.

BENEFITS OF THE RECONFIGURATION ALGORITHM

The use of simple, matrix-based graph representations of robots (for example, the adjacency matrix, incidence matrix, and difference matrix) is a key tool for automation of the reconfiguration process. Because the data structures involved are simple, and all of the algorithm steps can be accomplished using only these matrices, this representation lends itself well to self-reconfiguration as opposed to assisted reconfiguration.

The algorithm proposed here has several advantages over other reconfiguration algorithms. First, because it uses the difference matrix D, the first attempts at reconfiguration steps are aimed at those operations which are most efficient, or lead most directly toward the goal configuration.

Second, the algorithm allows for parallel execution of the reconfiguration steps. This is made possible through the technique of "virtual connection," which enables the algorithm to find a set of operations which can feasibly be done at the same time. Many reconfiguration algorithms based solely in graph theory are serial. While parallelization of serial algorithms is possible, it may not be the best solution or the most efficient. It is certainly an easier and more direct approach to begin with an algorithm which by its nature allows reconfiguration steps to be performed in parallel.

The final and most novel advantage of the proposed strategy is the incorporation of elements of the robot kinematics in the choice of reconfiguration steps. Because real-life robotic systems are highly constrained by their kinematics, a purely graph-theoretic approach is not ideal for the reconfiguration problem. This type of approach would lead to inefficiency due to selected reconfiguration steps being eliminated in the path-planning stage because of infeasible kinematics. Incorporation of the robot sub-chain workspace in the choice of reconfiguration steps reduces the load of path-planning calculations at a later stage, and greatly streamlines the process.

SUMMARY AND CONCLUSIONS

The strategy presented in this paper is an improvement over existing methods of self-reconfiguration in chain-type unit-modular robotic systems. Its consideration of the robot kinematics in the choice of reconfiguration steps makes it more efficient, as does its ability to perform multiple operations in parallel.

In addition to the reconfiguration algorithm itself, a new graph-theoretic approach to determining a robot's "mechanical composition" is summarized, along with a solution strategy for the preliminary subgraph-matching problem.

The discussion in this paper is limited to planar robotic link modules with a single rotational degree of freedom. Future work may include an extension of these methods to three dimensions, as well as adaptation for robotic modules having more (or different types of) degrees of freedom. Also remaining to be explored are incorporation of static-force and dynamic effects as criteria in the choice of reconfiguration steps, the effects of using different criteria for determining advantageous temporary disconnect operations, and the possibility of using the robot composition information in simplified mobility and kinematic analyses, as well as more in-depth experimentation into computer implementation of the proposed algorithm.

ACKNOWLEDGMENTS

While the content of this paper reflects uniquely the views of the authors, they wish to thank the National Science Foundation for supporting this research.

REFERENCES

[1] Casal, A., and Yim, M., 1999. "Self-Reconfiguration Planning for a Class of Modular Robots," Proceedings of SPIE, **3839**:246-257.

[2] Pamecha et al., 1997. "Useful Metrics for Modular Robot Motion Planning," IEEE Transactions on Robotics and Automation, **13**:531-545.

[3] Yoshida, E., et al., 1998. "Distributed Reconfiguration Method for 3-D Homogeneous Structure," IEEE International Conference on Intelligent Robots and Systems, **2**:852-859.

[4] Vassilvitskii, S., et al., 2002. "A Complete, Local and Parallel Reconfiguration Algorithm for Cube Style Modular Robots," IEEE International Conference on Robotics and Automation, **1**:117-122.

[5] Butler, Z., et al., 2002. "Distributed Replication Algorithms for Self-Reconfiguring Modular Robots," Distributed Autonomous Robotic Systems, **5**:37-48.

[6] Yim, M., et al., 2001. "Distributed Control for 3D Metamorphosis," Autonomous Robots, **10**:41-56.

[7] Bojinov, H., et al., 2000. "Emergent Structures in Modular Self-Reconfigurable Robots," IEEE International Conference on Robotics and Automation, **2**:1734-1741.

[8] Kotay, K., et al., 2000. "Using Modular Self-Reconfiguring Robots for Locomotion," Proceedings of Experimental Robotics IV, 259-269.

[9] Nguyen, A., et al., 2000. "Controlled Module Density Helps Reconfiguration Planning," Fourth International Workshop on Algorithmic Foundations of Robotics.

[10] Yim, M., et al., 2000. "Connectivity Planning for Closed-Chain Reconfiguration," Proceedings of SPIE, **4196**:402-412.

[11] Durna, M., et al., 2000(a). "The Self-Reconfiguration of a Holonic Hand: The Holonic Regrasp," IEEE International Conference on Intelligent Robots and Systems, **3**:1993-1998.

[12] Durna, M., et al., 2000(b). "Self-Reconfiguration in Task Space of a Holonic Structure," IEEE International Conference on Intelligent Robots and Systems, **3**:2366-2373.

[13] Sheth, P. N., and Uicker, J. J. Jr., 1972. "IMP (Integrated Mechanisms Program), A Computer-Aided Design Analysis System for Mechanisms and Linkage," ASME Journal of Engineering for Industry, May 1972.

[14] Smith, D. A., 1973. "Reaction Force Analysis in Generalized Machine Systems," ASME Journal of Engineering for Industry, May 1973.

[15] McGregor, J. J., 1982. "Backtrack Search Algorithms and the Maximal Common Subgraph Problem," Software – Practice and Experience, **12**:23-34.