Research on self-reconfiguration strategy of modular spherical robot

Hanxu Sun, Mingzhe Li, Jingzhou Song and Yun Wang

Abstract
Self-reconfigurable robot is a complex system composed of multiple modular robots. Aiming at high efficiency and low energy consumption of self-reconfigurable robot configuration transformation, a self-reconfiguration strategy based on module mapping of the common parts is proposed. This strategy describes the configuration of the robot in the form of a graph, and a method to determine the central node of configuration is proposed. The central node module as the starting node for comparison of different configurations, and the common part between the two configurations is reserved. Then the module closest to the target module is searched, the target configuration is reconfigured from the inside to the outside with the minimum energy consumption constraint. Finally, based on the experiment results, compared with other self-reconfiguration strategies, the proposed self-reconfiguration strategy reduces the times of reconfiguration operations and improves the reconfiguration efficiency.

Keywords
Self-reconfigurable robot, module mapping, common parts, self-reconfiguration strategy

Introduction
The modular self-reconfigurable robot is composed of several modules with connectors. When faced with an unstructured dynamic unknown environment, it can reconfigure different configurations through a series of operations such as moving, disconnecting, and connecting, so as to adapt to different environments and perform different tasks. Modular self-reconfigurable robots are mainly divided into three categories according to their topological structure: (1) lattice type: Most of the modules are cubes. In the configuration, each module is on the regular grid. Changing the position of the module, the configuration changes accordingly, such as Melecubes, Roombots, M-Cube, Cebot, and so on. (2) Chain type: The modules are connected in series to form a chain type, such as CONRO, Xmobot, USS, PolyBot, and so on. (3) Hybrid type: This module not only has good mobility capability but also has the structure characteristics of lattice and chain, such as M-TRAN, SuperBot, SamBot, SMORES, and so on.

Aiming at the transformation between different configurations of self-reconfigurable robot, Casal and Yim proposed the strategy of Divide-and-Conquer to match configurations in a layered manner, which involves many redundant reconfiguration steps and leads to low reconfiguration efficiency. Butler et al. proposed the strategy of self-reconfiguration of Divide-and-Conquer to match configurations in a layered manner, which involves many redundant reconfiguration steps and leads to low reconfiguration efficiency.
proposed a Melt-Group self-reconfiguration algorithm based on Crystalline module. This algorithm uses centralized control, which gradually "grows" the target configuration from a simple configuration. The centralized control method will reduce the efficiency of self-reconfiguration as the number of modules increases. Stoy et al.\textsuperscript{31} designed a method to transform configuration based on the transfer of hormone signals between modules to stimulate other hormones and generate corresponding reconfiguration action. This method has a single applicability, and the signal strength will be weakened in the transmission process, affecting the reconfiguration operation. Chao Liu\textsuperscript{32,33} proposed a self-reconfiguration strategy based on configuration decomposition for SMORES modules. By mapping two configurations with the same number of modules, a distributed reconfiguration operation was developed. This method did not specify the mapping starting point and did not take energy consumption into account, which made the reconfiguration efficiency low. The above self-reconfiguration strategies are all only suit for two configurations with the same total number of modules, but two configurations with different total number of modules have not been studied.

To improve the efficiency of self-reconfiguration for modular spherical robot, this article proposes a self-reconfiguration strategy based on module mapping of the common parts, which can reduce the energy consumption of configuration transformation by retaining common parts between the two configurations, shorten the moving distance of modules during the reconfiguration process and improve the efficiency of self-reconfiguration.

The structure of the article is as follows: The second section introduces the hardware structure of the spherical module; the third section introduces the configuration representation method of the self-reconfigurable robot based on the spherical module; the fourth section proposes the self-reconfiguration strategy, including determination of the central node module of the configuration, module mapping of common parts, and reconfiguration operation planning; the fifth section gives some experiments and results; and in the sixth section, the conclusions and future work are given.

**Spherical module**

The self-reconfigurable robot system studied in this article takes the connectable and openable spherical robot as the basic module, which is a hybrid-type modular robot. It combines the design concept of spherical robot and modular robot to solve the problem of poor motion ability of traditional modular robot. As shown in Figure 1, it is an evolutionary version of the robot designed in Ma et al.,\textsuperscript{34} there are six retractable and rotatable connectors distributed on the spherical module. Three of them are active connectors (connector 1, 3, 5) and other three connectors are passive connectors (connector 2, 4, 6), and connector 1 and connector 2 can swing up and down. The connection between connectors mainly depends on mechanism (latch structure), while electromagnetic devices are installed to assist the connection and ensure the reliability of the connection between the two modules. Considering the communication requirement between modules, there are some communication interfaces on the connectors of the module.

The spherical module can not only perform tasks by rolling alone but also connect with other modules to form different configurations. The spherical module moves in two ways: (1) Spherical shell closure: the movement of the spherical module is realized by deviation of mass center, and the movement of the mass block inside the module can change the mass center of the spherical module. When the mass block is driven forward by the long axis motor, the spherical module starts to move forward. When the short axis motor drives the mass block to the left and right, the spherical module turns to the corresponding direction, as shown in Figure 2(a). (2) Spherical shell opening: The spherical module becomes a wheel-type robot, and the left and right hemispherical shells are equivalent to the left and right wheels of the wheel-type robot, which can realize differential motion and are mainly used for the connecting motion between modules, as shown in Figure 2(b).

All connection modes between unit modules are listed in Figure 3. Connection modes (a) to (h) can be butted through the proximity of two spherical unit modules and the expansion and contraction of connectors. However, due to the limitation of the unit module itself and the extension length of the connector, the connection mode of (i) will cause interference when the two spherical unit module connectors 5 and 6 are connected. Therefore, there are only eight connection modes (a) to (i) in the figure between spherical unit modules. And among the eight connection modes, (e) requires two spherical unit modules to be close to each other, so as to provide a horizontal coupling force for the connection between them to complete the docking smoothly, while all other connection modes can be moved from one spherical unit module to the vicinity of another spherical unit module, and their connectors can be aligned to complete the connection.

In the connection relationship (a) to (h) in Figure 3, not every connected spherical unit module can move flexibly. Among the four connection relationships, such as (b), (d), (f), and (h), due to the rigid connection between the two modules and the limitation of the weight of the spherical unit module, the two modules cannot move flexibly under these four connection modes. Under the three connection relationships, such as (a), (c), and (g), the long axis motor of one spherical unit module can be powered off, and the connected other spherical unit module can drive it to move, just like a four-wheel vehicle. In this way, such as (e), the two hemispherical shells close to the two connected unit
Figure 1. The structure of spherical module. (a) The structure of spherical module with spherical shell. (b) The structure of spherical module without spherical shell.

Figure 2. The two motion modes of spherical module. (a) Spherical shell closure. (b) Spherical shell open.
modules follow, and the other two hemispherical shells are equivalent to the two wheels of a two-wheeled vehicle. The module in this connection mode can still move flexibly and stably like a two-wheeled vehicle.

### Configuration representation

The configuration representation of self-reconfigurable robot is the basis of the self-reconfiguration strategies. Combining the characteristics of the spherical module with the knowledge of graphics, the spherical module can be simplified to a node, as shown in Figure 4. $M_i$ denotes the number of the module in the self-reconfigurable robot system, and connection line denotes the connectors distributed on the spherical module. The digital 1 to 6 corresponds to the number of the connector, where the odd numbers indicate the active connector and the even numbers indicate the passive connector. Only the connection between the active connector and the passive connector is an effective connection.

The self-reconfigurable robot is composed of several modules and can be described by the directed topology diagram in the graph theory. Let $G = (V, E)$ represents a configuration of the robot, where $V = \{M_1, M_2, M_3, \ldots, M_i\}$ represents the set of all node modules of the configuration $G$, and $E = \{e_{1,2}, e_{2,3}, e_{3,4}, \ldots, e_{ij}\}$ represents the set of edges connected between two modules of the configuration $G$. As shown in Figure 5, it is a graphical representation of snake configuration. In Figure 6, it is a graphical representation of planar quadruped configuration (the representation of spatial quadruped configuration is the same as that of planar quadruped configuration). The $e_{ij} = (c_n, c_m)$ in $E$ represents the connecting edge between module $M_i$ and module $M_j$, where $c_n$ and $c_m$ represent the connectors used for the connection of the two modules. For example as shown in Figure 5, $e_{3,4} = (1, 2)$ indicates that $c_1$ of $M_3$ is connected to $c_2$ of module $M_4$.

Given two configurations $G_1 = (V_1, E_1)$ and $G'_1 = (V'_1, E'_1)$ of self-reconfigurable robot, the common part of the two configurations is defined as: if $e_{ij} \in E_1 = e'_{p,q} \in E'_1$
exists in configuration $G_i$ and configuration $T_i$, there is a common part of the two configurations and can be expressed as $e_{ij}^{G_i} = e_{ij}^{T_i}$. An example in Figure 7, the common parts of the two configurations are as follows: $e_{1,2}^{G_i} = e_{1,2}^{T_i}$ or $e_{2,3}^{G_i} = e_{1,2}^{T_i}$ or $e_{3,4}^{G_i} = e_{1,2}^{T_i}$.

Commonly used nomenclature is listed in the following.

- $G_i$: Initial configuration;
- $T_i$: Target configuration;
- $M_j$: The central node of the initial configuration;
- $M'_j$: The central node of the target configuration;
- $CN_j[i]$: The count of modules connected by each connector $c_i$ of $M_j$ (including direct and indirect connected), expressed in the form of array;
- $\text{CN}_j[i]$: It is obtained by sorting the elements in array $CN_j[i]$ from max to min;
- $Ns$: Total number of modules included in a configuration;
- $S_f[i]$: The number of modules connected by all connectors (except connector $c_i$) of $M_j$.

**Self-reconfiguration strategy**

The self-reconfiguration strategy is defined as: based on the given two different configurations $G_i$ and $T_i$, a strategy is formulated to reduce the transformation steps from configuration $G_i$ to configuration $T_i$ and decrease energy consumption. The self-reconfiguration operation of most robots consists of disconnecting, moving, and connecting. Connection operation requires two modules to align and then activate the relevant connecting mechanism. Correspondingly, a disconnection operation invalidates the connection of the connector involved and separates the two modules. This article not only decreases the energy consumption of connection and disconnection but also considers minimizing the energy consumption in the moving process.

The self-reconfiguration strategy can be divided into three steps: First, the central node module of the configuration is determined. Then, the central node module is used as the starting node of configuration comparison to obtain the common parts of the two configurations. Finally, the common parts move to the corresponding position by differential motion according to the target configuration information and the noncommon parts are reconfigured.

**Central node module**

In any configuration of self-reconfigurable robot, the array $CN_j$ shows the distribution of modules around $M_j$. For example as in Figure 6, $CN_9[3] = 3$, it represents there are three modules connected with $c_3$ of $M_6$ in the planar
quadruped configuration (i.e. $M_{10}$, $M_{11}$, $M_{12}$). And the sum of the elements in the $CN$ of any module is equal to $N_s - 1$. Therefore, when the number of nonzero elements in the module $CN$ is greater than one. Suppose $num 1$ is the largest element in $CN^{i,j}$ and $num 2$ is the largest element in $CN^{i,j}$. If $num 1 > num 2$, it means that the modules around $M_1$ are evenly distributed, then $M_{j1}$ is the central node module of the configuration.

In any configuration of a self-reconfigurable robot, all modules remain connected, and each module communicates with the connected modules through its own $CN$ through a communication interface. Assuming that the connector $c_{in}$ of the $M_1$ is connected to the connector $c_{in}$ of the $M_2$, we can determine $CN^{M_1}[c_{in}]$. Firstly, the $CN[i]$ of each connector in $M_2$ (except connector $c_{in}$) is determined. Then, the $S^n_{i}$ of $M_2$ is calculated by equation (1). Finally, $S^n_{i}$ is passed to $M_1$ and $CN^{M_1}[c_{in}] = S^n_{i}+1$. In the whole self-reconfigurable robot system, if a node module with only one connector connected to other modules is called leaf node module. Therefore, the process of determining $CN$ can be started by either leaf node module until $CN$ of all modules in the entire robotic system are determined. For example in Figure 6, connector $c_1$ of the $M_6$ is connected to connector $c_2$ of the $M_9$, for the $M_9$, $CN^{M_9}[1] = 0$, $CN^{M_9}[3] = 3$, $CN^{M_9}[4] = 3$, $CN^{M_9}[5] = 0$, $CN^{M_9}[6] = 0$, then the $CN^{M_9}[c_1] = S^{M_9} + 1 = 3 + 3 + 1 = 7$ can be obtained from equation (1)

$$S^n_{i} = \sum_{n \neq i}^{6} CN^n[n]$$

where $j$ is the module, $i$ is the connector connected with other modules, $n$ is all connector of the $M_j$.

The $CN$ of all modules in the current configuration can be determined by the above method. Since there is only one nonzero element in $CN$ of leaf node module, and the leaf node module cannot become the central node module of the configuration. Therefore, the modules with the number of nonzero elements greater than or equal to 2 in the $CN$ are recorded. If the recorded module is unique, this module is the central node module of the current configuration. If they are not unique, then the elements in $CN$ are sorted in descending order to get $CN$. Next, if the first element in $CN$ is the minimum value and the module is unique, this module is the central node module of the configuration; if it is not unique, continue to compare the second elements in $CN$ to find the minimum value, until find the qualified module as the central node module of the current configuration through iteration. If more than one module is found at the same time, then select any one as the central node module of the configuration. The algorithm pseudo code of this process is as in Algorithm 1:

Algorithm 1. Search central node module.

1: Input: $CN'[i]$ : The number of module connected by connector $c_i$ of $M_j$.
2: $N_s$ : The total number of modules in the configuration;
3: Output: $M$ : The central node module in the configuration;
4: Count the number of non-zero elements in $CN$ of each module;
5: $Res$ = Record modules with two or more nonzero elements in $CN$;
6: End
7: If $Res$ only has one module($M_j$) then
8: $M = M_j$ ; break;
9: Else
10: $CN = \text{Sorting all elements from max to min in } CN$ of all modules in $Res$;
11: End
12: For each $n \in [1,6]$ do
13: If $CN[n]$ is the smallest in the $CN$ then
14: $M = M_j$ ; break;
15: Else if $CN'[n] = CN'[j]$ is the smallest in
16: the $CN$ then
17: $M = M_{j1}$ or $M_{j2}$ ;
18: End

Table 1. The $CN$ and $\bar{CN}$ of snake configuration.

| Module $M_j$ | $CN$ | $\bar{CN}$ |
|--------------|------|-------------|
| 1            | [6,0,0,0,0] |             |
| 2            | [5,1,0,0,0] | [5,1,0,0,0] |
| 3            | [4,2,0,0,0] | [4,2,0,0,0] |
| 4            | [3,3,0,0,0] | [3,3,0,0,0] |
| 5            | [2,4,0,0,0] | [4,2,0,0,0] |
| 6            | [1,5,0,0,0] | [5,1,0,0,0] |
| 7            | [0,6,0,0,0] |             |

$\bar{CN}$ of planar quadruped configuration. The module $M_4$ is finally determined as the central node module of planar quadruped configuration.

Module mapping of common parts

The self-reconfiguration process of modular robot is that the robot transforms from the initial configuration to the target configuration according to the characteristics of the environment. In this process, the initial configuration and the target configuration should be analyzed and compared to find out the common parts and noncommon parts...
between the two configurations. The existing connection between modules in the initial configuration should be retained as much as possible in the target configuration so as to reduce the energy and time consumption of disconnection and connection operations, and improve the reconfiguration efficiency.

The energy consumption of each self-reconfiguration operation from \( G \) to \( T \) can be defined as:

\[
\text{cost}_{\text{mov}}(M_i, M_j) \quad \text{represents the energy required to move module } M_i \text{ in } G \text{ to module } M_j \text{ in } T;
\]

\[
\text{cost}_{\text{con}}(M_i, M_j) \quad \text{represents the energy required to connect module } M_i \text{ and module } M_j;
\]

\[
\text{cost}_{\text{dis}}(M_i, M_j) \quad \text{represents the energy required to disconnect module } M_i \text{ from module } M_j.
\]

Then the total energy consumed by the robot from \( G \) to \( T \) is as follows:

\[
\text{cost}(G, T) = \sum \text{cost}_{\text{dis}}(M_i, M_j) + \sum \text{cost}_{\text{mov}}(M_i, M_j) + \sum \text{cost}_{\text{con}}(M_i, M_j)
\]

(2)

In order to improve the efficiency of reconfiguration and reduce the energy consumption in the process of self-reconfiguration, firstly, the initial configuration and the target configuration are compared to obtain all the common parts of the two configurations, and the connection of the common parts is retained to reduce the times of disconnection and connection operations in the self-reconfiguration process, and ensure that \( \sum \text{cost}_{\text{con}}(M_i, M_j) \) and \( \sum \text{cost}_{\text{dis}}(M_i, M_j) \) are the minimum.

Based on the given initial configuration \( G = (V, E) \) and target configuration \( T = (V', E') \), according to the method in the “Central node module” section, the central node module \( M_c \) of the initial configuration and target configuration are determined, respectively. Starting from the \( M_c \) of the initial configuration \( G \), and using depth first search algorithm, we can obtain all the branch chains \( G_1, \ldots, G_k \) of the initial configuration \( G \). (In any configuration, from the central node module to the leaf node module is a branch chain.) In the snake configuration shown in Figure 5, the central node module is \( M_3 \), and there are two branch chains \( G_1 = \{V_1, E_1\} \) and \( G_2 = \{V_2, E_2\} \), \( V_1 = \{4, 5, 6, 7\} \) and \( V_2 = \{4, 3, 2, 1\} \), \( E_1 = \{e_{4,5}(1, 2), e_{5,6}(1, 2), e_{6,7}(1, 2)\} \) and \( E_2 = \{e_{1,2}(1, 2), e_{2,3}(1, 2), e_{3,4}(1, 2)\} \). Meanwhile, starting from the central node \( M_c' \) of the target configuration \( T \), all its branch chains \( T_1 \ldots T_l \) also are obtained. The planar quadruadrated configuration we can see that in Figure 6 has a central node module is \( M_8 \) and four branch chains \( T_1 = \{V_1', E_1'\} \), \( T_2 = \{V_2', E_2'\} \), \( T_3 = \{V_3', E_3'\} \), and \( T_4 = \{V_4', E_4'\} \); \( V_1' = \{8, 9, 10, 11, 12\} \), \( V_2' = \{8, 9, 13, 14, 15\} \), \( V_3' = \{8, 4, 3, 2, 1\} \), and \( V_4' = \{8, 4, 5, 6, 7\} \); \( E_1' = \{e_{8,9}(1, 2), e_{9,10}(3, 2), e_{10,11}(3, 2), e_{11,12}(1, 2)\} \), \( E_2' = \{e_{8,9}(1, 2), e_{9,13}(4, 1), e_{13,14}(3, 2), e_{14,15}(1, 2)\} \), \( E_3' = \{e_{4,8}(1, 2), e_{3,4}(2, 3), e_{2,3}(1, 4), e_{1,2}(1, 2)\} \), and \( E_4' = \{e_{4,8}(1, 2), e_{4,5}(4, 1), e_{5,6}(4, 1), e_{6,7}(2, 1)\} \). Next, the connection information of each branch chain of the initial configuration is compared with that of all branch chains in the target configuration. Then all edges with the same connection in the initial configuration \( G \) and the target configuration \( T \) are obtained, then mapping modules of the common parts.

Note that there are several conflicts in the reconfiguration need to consider:

1. The \( M_c \) of initial configuration is not the \( M_c' \) of target configuration after reconfiguration.
2. Multiple edges of the initial configuration have the same connection information with one edge of the target configuration.
3. One edge of the initial configuration has the same connection information with multiple edges of the target configuration.
4. The adjacent edges in the initial configuration have the same connection information with the nonadjacent edges in the target configuration.
5. The adjacent edges in the target configuration have the same connection information with the nonadjacent edges in the initial configuration.

Therefore, all results should be filtered. For (1), the central node of the initial configuration that does not map to the central node of the target configuration is deleted from all results. For (2) and (3), the coordinates of the central node of the initial configuration are assigned to the central node of the target configuration. In the case of multiple edges have the same connection information with one edge, the distance between each edge and this edge is calculated respectively, and the edge with the shortest distance is selected to map with this edge (if there are several shortest edges, choose one of them), and
Algorithm 2. Get the common parts.

1: **Input:** \( G = (V, E) \) : Initial configuration;
   \( T = (V', E') \) : Target configuration;
2: **Output:** \( \text{Res} \) : The common parts between two configurations;
3: \( G_2 \) : The DFS algorithm is used to obtain all branched chains in the initial configuration;
4: \( T_2 \) : The DFS algorithm is used to obtain all branched chains in the target configuration;
5: **For** \( i = 1 \) to \( m \) **do**
6:   **For** \( j = 1 \) to \( n \) **do**
7:      \( \text{res} = G_{\cap} T_{\cap} \) (Parts with the same connection);
8: **End**
9: **End**
10: **If** central node does not map **then**
11:     \( \text{res}_s \) : delete central node does not map from \( \text{res} \);
12: **End**
13: **If** multiple edges map one edge **then**
14:     \( \text{res}_s \) : delete edges whose distance is farther away one edge from \( \text{res}_s \);
15: **End**
16: \( \text{Res} = \text{res}_s \);

the others are deleted from the result. For (4) and (5), the distance between the two mapping edges is also calculated, and the pair of mapping edges with the shortest distance is retained. (If more than one edge meets the condition, select any one.) In this way, the remaining mapping edges are all the common parts of configuration \( G \) and configuration \( T \). The algorithm pseudo code of this process is as seen in Algorithm 2.

Reconfiguration operating

The process of transforming self-reconfigurable robot from initial configuration \( G \) to target configuration \( T \) follows the principle of disconnecting first and then connecting. First, disconnect the connection of noncommon parts in initial configuration \( G \). Then, according to the connection information of target configuration \( T \), the module performs the reconfiguration from the inside to the outside. This can avoid locking due to the occupancy conflict of the module during the reconfiguration process and make other modules unable to connect with the module.

After mapping the initial configuration \( G \) with the target configuration \( T \), the noncommon part of the initial configuration \( G \) is separated according to the minimum times of disconnections. The common part of initial configuration \( G \) which is the same as target configuration \( T \) moves to the corresponding position of target configuration \( T \) through differential motion, as shown in Figure 8(b). Next, the reconfiguration operation is conducted for the noncommon parts. According to the connection information of target configuration \( T \), the central node module of initial configuration \( G \) or the common part containing the central node module is taken as the reconfiguration center, and the number of the target module in the current configuration is determined, as shown in Figure 8(c). Then, the target modules broadcast their location information one by one in order, all the surrounding modules calculate the distance from the modules to their own the target position, and select several modules closest to the target modules in the current configuration (satisfying the number of connections required by the target modules), as shown in Figure 8(d). Each module uses the planning algorithm (such as RRT, etc.) to generate the feasible path to the target module according to the environmental information and its own state, and conduct the orientation transformation to connect with the target module. In this way, the position of the target module in the current configuration is broadcast, and the nearest module is determined. Feasible path is generated to ensure the safe and orderly movement of the module near the module to be connected to complete the connection operation, and finally implement the target configuration \( T \), as shown in Figure 8(e), (f), and (g). The reconfiguration process from snake configuration to planar quadruped configuration is shown in Figure 8.

The self-reconfiguration strategy takes into account the energy consumption during the transition from initial configuration to target configuration. By comparing the connection information of initial configuration and target configuration, the common parts of them are retained, thus reducing the energy consumption caused by disconnection \( \sum \text{cost}_{\text{dis}}(M_i, M_j) \) and connection \( \sum \text{cost}_{\text{con}}(M_i, M_j) \). Meanwhile, the distance between the initial configuration and the common part of the target configuration is reduced as much as possible so as to reduce the energy loss in the process of part moving \( \sum \text{cost}_{\text{mov}}(M_i, M_j) \). In the process of reconfiguration, we will find the connectable module that is nearest to the target module in the current configuration and generate a feasible path for it so as to reduce the energy consumption \( \sum \text{cost}_{\text{mov}}(M_i, M_j) \). Through obtaining common parts and reconfiguration operations, the total energy consumption is minimized, thus reducing the energy consumption and improving the efficiency of reconfiguration.

Experiments and results

We have implement the algorithm on a desktop PC (Intel Core i5-6500, 3.20 GHz, 4 GB SDRAM). We perform two types of experiments. First, we reconfigure two
configurations with different total number of modules to verify the feasibility of the algorithm. Second, we reconfigure two configurations with same total number of modules to verify the superiority of the algorithm compared with other algorithms.

The total number of modules is different
In the existing self-reconfiguration strategies, such as Hou et al., most self-reconfiguration strategies are to find the part with the most same connection relationship between the initial configuration and the target configuration, keep it unchanged in the initial configuration, then reconfigure other parts, and they are only applicable to the case that the total number of modules in the initial configuration and the target configuration are the same. This article is looking for the part with the most same connection relationship between the initial configuration and the target configuration and considers reducing the energy consumption in the transformation process. Therefore, the self-

---

**Figure 8.** Reconfiguration process from snake configuration to planar quadruped configuration. (a) The common parts between the G and T. (b) Move common parts. (c) The target modules of the first layer. (d) Modules close to the target modules of the first layer. (e) Connect to the target module and the target modules of the second layer. (f) Connect to the target module and the target modules of the third layer. (g) The target configuration.
reconfiguration strategy proposed in this article considers the energy loss on its basis, so it has certain advantages.

In this section, we completed two reconfiguration tasks to verify the feasibility of the self-reconfiguration strategy. First, we completed the reconfiguration task from snake configuration to planar quadruped configuration. (The total number of modules in the target configuration is larger than that in the initial configuration.) The common parts of snake configuration and planar quadruped configuration are shown in Figure 9 and represented by dotted lines. The combination of digit and dot is used to represent the module and connection line represents the connection between two modules, the digit on the connection line corresponds to the connection relationship between the two modules as shown in Table 3. For example, in the snake configuration of Figure 9, the digit 1 on the connection line between $M_1$ and $M_2$ indicates the connection between $c_1$ of $M_1$ and $c_2$ of $M_2$ (or the connection between $c_1$ of $M_1$ and $c_1$ of $M_2$). The central node of snake configuration is $M_4$, the central node of planar quadruped configuration is $M_8$. Snake configuration has two branched chains $G_1$, $G_2$, planar quadruped configuration has four branched chains $T_1$, $T_2$, $T_3$, $T_4$. The results of edges with the same connection relationship are shown in Table 4. After filtering, the results of module mapping of the common parts are $e_{1,2}^G = e_{1,2}^T$, $e_{3,4}^G = e_{4,5}^T = e_{8,9}^T = e_{6,7}^T = e_{11,12}^T$. The reconfiguration process is shown in Figure 8. The snake configuration disconnects the edges $e_{2,3}$ and $e_{5,6}$. Based on the connection information of the planar quadruped configuration, the $M_1, M_4, M_5$ which maintain the connection of the snake configuration are taken as the reconfiguration center, while the rest of the common parts move to the corresponding position in a differential motion manner. The scattered modules around them move orderly to the target modules to in the reconfiguration center one by one to perform the reconfiguration operation, thus forming the planar quadruped configuration. Because the graphic representation of spatial configuration is the same as that of planar configuration. If the target configuration is a spatial quadruped configuration, the robot can first reconfiguration its corresponding planar quadruped configuration, then gradually adjust to form the spatial quadruped configuration.

Second, we completed the reconfiguration task from windmill configuration to H-shape configuration. (The total number of modules in the initial configuration is larger than that in the target configuration.) The common parts of snake configuration and planar quadruped configuration are shown in Figure 10 and represented by dotted lines. The central node of windmill configuration is $M_3$, and the central node of H-shaped configuration is $M_4$.
The reconfiguration process is shown in Figure 14. The H-shape configuration and snake configuration are shown in Figure 12 and 13. The windmill configuration disconnects the edges $e_{3,4}$ and $e_{3,7}$. According to the connection information of H-shaped configuration, the $M_1, M_2, M_3, M_6,$ and $M_9$ which maintain the connection of the windmill configuration are taken as the reconfiguration center, and the scattered modules move orderly to the target modules in the reconfiguration center one by one, forming the H-shaped configuration.

The total number of modules is same

In this section, we completed two reconfiguration tasks and compared them with other reconfiguration strategies to verify the superiority of the strategy. First, we completed the reconfiguration task from H-shaped configuration to snake configuration. The common parts of H-shaped configuration and snake configuration are shown in Figure 12 and represented by dotted lines. The center node of H-shaped configuration is $M_4$, and that of snake configuration also is $M_4$. The result of module mapping of the common parts is $e_{1,2}^G = e_{1,2}^T$, $e_{2,3}^G = e_{2,3}^T$, $e_{5,6}^G = e_{5,6}^T$, $e_{4,5}^G = e_{4,5}^T$. The reconfiguration process is shown in Figure 13. The H-shape configuration disconnects edge $e_{2,4}$ and edge $e_{4,5}$, moves related modules $(M_1, M_2, M_3, M_6, M_5, M_7)$, and connects $M_3$ and $M_4$ to form edge $e_{3,4}$, connects $M_4$ and $M_6$ to form edge $e_{4,6}$. However, the results of module mapping of self-reconfiguration strategy of Chao Liu are $1 \rightarrow 7, 2 \rightarrow 5, 3 \rightarrow 6, 4 \rightarrow 4, 5 \rightarrow 3, 6 \rightarrow 2,$ and $7 \rightarrow 1$. The reconfiguration process is shown in Figure 14. The H-shape configuration disconnects these edges which are $e_{1,2}, e_{7,5}, e_{2,4},$ and $e_{3,4}$, moves related modules $(M_1, M_2, M_3, M_6, M_5, M_7)$, connects $M_1$ and $M_3$ to form edge $e_{1,3}$, connects $M_7$ and $M_6$ to form edge $e_{7,6}$, connects $M_5$ and $M_4$ to form edge $e_{2,4}$, and connects $M_5$ and $M_4$ to form edge $e_{5,4}$.

In Dutta et al., the energy consumption of self-reconfigurable robot in configuration transformation is defined, but there is no quantitative analysis. In order to facilitate comparison with other self-reconfiguration strategies, this article uses the method of defining its energy
consumption for reference, and makes quantitative assumptions on its energy consumption, as follows:

Since the movement of the unit module requires continuous power supply to the motor, therefore the energy consumption is much higher than that of connection. In addition, the connection process needs to align the connector first, which is higher than the energy consumption of disconnecting two modules. Therefore, it is assumed that:

\[
\begin{align*}
\text{cost}_{\text{dis}}(M_i, M_j) &= 1 \\
\text{cost}_{\text{con}}(M_i, M_j) &= 2 \\
\text{cost}_{\text{mov}}(M_i, M_j) &= \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + 2
\end{align*}
\]

where \( M_i = (x_i, y_i) \) and \( M_j = (x_j, y_j) \) are the coordinates of the modules in Figure 12.

By comparing the two self-reconfiguration strategies, the energy consumption of the robot guided by the self-reconfiguration strategy proposed in this article is obtained as follows:

\[
\begin{align*}
\text{cost}(G, T) &= \sum \text{cost}_{\text{dis}} + \sum \text{cost}_{\text{con}} + \sum \text{cost}_{\text{mov}} \\
&= (1 + 1) + (2 + 2) + (14 + 6\sqrt{5}) \\
&\approx 33.416
\end{align*}
\]

The energy consumption of the robot guided by Chao Liu’s self-reconfiguration strategy is as follows:

\[
\begin{align*}
\text{cost}'(G, T) &= \sum \text{cost}_{\text{dis}} + \sum \text{cost}_{\text{con}} + \sum \text{cost}_{\text{mov}} \\
&= (1 + 1 + 1 + 1) + (2 + 2 + 2 + 2) + (14 + 4\sqrt{2} + 2\sqrt{17}) \\
&\approx 39.903
\end{align*}
\]

In conclusion, \( \text{cost}(G, T) < \text{cost}'(G, T) \), it shows that the self-reconfiguration strategy proposed in this article

---

**Figure 12.** The common parts of H-shape configuration and snake configuration.

**Figure 13.** Reconfiguration process from H-shape configuration to snake configuration by this article’s strategy.
reduces the energy consumption by 16.25% and improves the efficiency of self-reconfiguration.

Second, we completed the reconfiguration task from A configuration to B configuration. The common parts of A configuration and B configuration are shown in Figure 15 and represented by dotted lines. The center node of A configuration is $M_1$, and that of B configuration also is $M_1$. The result of module mapping of the common parts are $e_{1,2}^G = e_{1,8}^T$, $e_{2,3}^G = e_{6,9}^T$, $e_{4,5}^G = e_{6,3}^T$, $e_{8,9}^G = e_{4,2}^T$. The reconfiguration process is shown in Figure 16. The A configuration needs to disconnect edge $e_{1,4}$, edge $e_{1,8}$, edge $e_{1,6}$, and edge $e_{6,7}$, move related modules ($M_4$, $M_5$, $M_6$, $M_7$, $M_8$, $M_9$), connect $M_9$ and $M_1$ to form edge $e_{9,1}$, connect $M_5$ and $M_1$ to form edge $e_{5,1}$, connect $M_6$ and $M_9$ to form edge $e_{6,9}$, and connect $M_5$ and $M_7$ to form edge $e_{5,7}$. However, the results of module mapping of self-reconfiguration strategy of Chao Liu\textsuperscript{32} are $1 \rightarrow 1$, $2 \rightarrow 8$, $3 \rightarrow 9$, $4 \rightarrow 2$, $5 \rightarrow 5$, $6 \rightarrow 4$, $7 \rightarrow 6$, $8 \rightarrow 3$, and $9 \rightarrow 7$. The reconfiguration process is shown in Figure 17. The A configuration needs to disconnect edge $e_{1,4}$, edge $e_{1,8}$, edge $e_{1,6}$, and edge $e_{6,7}$, move related modules ($M_4$, $M_5$, $M_6$, $M_7$, $M_8$, $M_9$), connect $M_4$ and $M_1$ to form edge $e_{4,1}$, connect $M_8$ and $M_1$ to form edge $e_{8,1}$, connect $M_4$ and $M_4$ to form edge $e_{6,4}$ and connect $M_8$ and $M_7$ to form edge $e_{8,7}$.

The energy consumption of the robot guided by the self-reconfiguration strategy proposed in this article is obtained as follows:

![Figure 14. Reconfiguration process from H-shape configuration to snake configuration by Chao Liu’s strategy.](image)

![Figure 15. The common parts of A configuration and B configuration.](image)
Figure 16. Reconfiguration process from A configuration to B configuration by this article’s strategy.

Figure 17. Reconfiguration process from A configuration to B configuration by Chao Liu’s strategy.
cost(G, T) = \sum \text{cost}_{\text{dis}} + \sum \text{cost}_{\text{con}} + \sum \text{cost}_{\text{mov}}
= (1 + 1 + 1) + (2 + 2 + 2 + 1) + (17 + \sqrt{2})
\approx 30.414

The energy consumption of the robot guided by Chao Liu’s self-reconfiguration strategy is as follows:

cost'(G, T) = \sum \text{cost}_{\text{dis}} + \sum \text{cost}_{\text{con}} + \sum \text{cost}_{\text{mov}}
= (1 + 1 + 1 + 1) + (2 + 2 + 2 + 1) + \ldots
= (12 + 3\sqrt{10} + \sqrt{5})
\approx 35.723

In conclusion, cost(G, T) < cost'(G, T), it shows that the self-reconfiguration strategy proposed in this article reduces the energy consumption by 14.86% and improve the efficiency of self-reconfiguration.

Conclusion

In this article, self-reconfiguration strategy based on module mapping of the common parts is proposed. The goal of this strategy is to minimize the energy consumption in the process of configuration transformation. The connection information between the initial configuration and the target configuration is fully compared to determine the common part between the two configurations. The existing connection between the modules in the initial configuration is retained in the target configuration as far as possible, so as to ensure minimum energy consumption for connection and disconnection during conversion process. Also in the process of reconfiguration, the module closest to the target module is used for connecting operation, which reduces the energy consumption in the process of moving, thus minimizing the energy consumption of the whole system and improving the efficiency of self-reconfiguration. Through the experiment of self-reconfiguration strategy, the results show that the self-reconfiguration strategy is effective and feasible, and the strategy is not only suitable for two configurations with the same number of modules but also suitable for two configurations with different number of modules. Future work will focus on physical experiments.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed the receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the Fundamental Research Funds for Central Universities under Grant 2019XD-A16.

ORCID iDs

Mingzhe Li https://orcid.org/0000-0003-1167-8629
Jingzhou Song https://orcid.org/0000-0002-7228-989X

Supplemental material

Supplemental material for this article is available online.

References

1. Yim M, Shen WM, Salemi B, et al. Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robot Autom Mag 2007; 14(1): 43–52.
2. Yim M, White P, Park M, et al. Modular self-reconfigurable robots. In: Robert N. Meyers (ed). Encyclopedia of complexity and systems science 2009; 5618–5631. New York: Springer.
3. Xu W, Wen-hua J, and Shi-gang W. The architecture and the classified study of self-reconfigurable robots. Manuf Autom 2003(07): 30–32.
4. Asadpour M, Sproewitz A, Billard A, et al. Graph signature for self-reconfiguration planning of modules with symmetry. In: IEEE/RSJ international conference on intelligent robots & systems, St. Louis, MO, USA, 10–15 October 2009. IEEE Press.
5. Yanqiong F and Wei Y. Structure description of a novel hybrid self-reconfigurable modular robot. J Shandong Univ Sci Technol (Natural Science Edition) 2012; 31(004): 51–55.
6. Zykov V, Mytilinaios E, Desnoyer M, et al. Evolved and designed self-reproducing modular robotics. IEEE Trans Robot 2007; 23(2): 308–319.
7. Bonardi S, Moeckel R, Sproewitz A, et al. Locomotion through reconfiguration based on motor primitives for Roombots self-reconfigurable modular robots. In: German conference on Robotics Robotik, Munich, Germany, 21–22 May 2012. VDE.
8. Sproewitz A, Pouya S, Bonardi S, et al. Roombots: reconfigurable robots for adaptive furniture. IEEE Comput Intellig Mag 2010; 5(3): 20–32.
9. Spröwitz A, Möckel R, Vespignani M, et al. Roombots: a hardware perspective on 3D self-reconfiguration and locomotion with a homogeneous modular robot. Robot Autonom Syst 2014; 62(7): 1016–1033.
10. Xia P, Zhu XJ, and Fei YQ. Design and metamorphosis method of a homogeneous and lattice self-reconfigurable robot. J Shanghai Jiaotong Univ 2006; 40(11): 1822–1826.
11. Fei Y, Zhu Y, and Xia P. Analysis on self-morphing process of self-reconfigurable modular robot. Int J Adv Robot Syst 2009; 6(3): 23.
12. Fukuda T and Nakagawa S. A dynamically reconfigurable robotic system (concept of a system and optimal configurations). In: Conference of the IEEE industrial electronics society, Philadelphia, PA, USA, 24–29 April 1987. IEEE.
13. Fukuda T, Buss M, Hosokai H, et al. Cell structured robotic system CEBOT: control, planning and communication methods. Robot Autonom Syst 1991; 7(2–3): 239–248.
14. Castano A, Behar A, and Will PM. The Conro modules for reconfigurable robots. IEEE/ASME Trans Mechatr 2003; 7(4): 403–409.
15. Rubenstein M, Payne K, Will P, et al. Docking among independent and autonomous CONRO self-reconfigurable robots.
16. Qiao G, Song G, Wang W, et al. Design and implementation of a modular self-reconfigurable robot. *Int J Adv Robot Syst* 2014; 11: 47.

17. Gao R Y, Wu C, and Ge T. Hardware design of an underwater self-reconfigurable robot. *Robot* 2007; 29(3): 239–249.

18. Yim M, Duff DG, and Rousas KD. PolyBot: a modular reconfigurable robot. In: *IEEE international conference on robotics & automation*, San Francisco, CA, USA, 24–28 April 2000. IEEE.

19. Golovinsky A, Yim M, Zhang Y, et al. PolyBot and PolyKinetic(TM) system. In: *IEEE international conference on robotics & automation*, New Orleans, USA, 23–27 May 2004. IEEE.

20. Kurokawa H, Kamimura A, Yoshida E, et al. M-TRAN II: metamorphosis from a four-legged walker to a caterpillar. In: *IEEE/RSJ international conference on intelligent robots & systems*, Las Vegas, NV, USA, 27–31 October 2003. IEEE.

21. Kurokawa H, Tomita K, Kamimura A, et al. Distributed metamorphosis control of a modular robotic system M-TRAN. In: *Distributed autonomous robotic systems 7*. Japan: Springer, 2006.

22. Kurokawa H, Tomita K, Kamimura A, et al. Distributed self-reconfiguration of M-TRAN III modular robotic system. *Int J Robot Res* 2008; 27(3–4): 373–386.

23. Shen WM, Krivokon M, Chiu HCH, et al. Multimode locomotion via SuperBot robots. In: *Proceedings 2006 IEEE international conference on robotics and automation*, 2006, *ICRA* 2006, Orlando, FL, USA, 15–19 May 2006. IEEE.

24. Salemi B, Moll M, and Shen WM. SUPERBOT: a deployable, multi-functional, and modular self-reconfigurable robotic system. In: *IEEE/RSJ international conference on intelligent robots & systems*, Beijing, China, 9–15 October 2007. IEEE.

25. Shen WM, Chiu HCH, Rubenstein M, et al. Rolling and climbing by the multifunctional superbot reconfigurable robotic system. *AIP Conf Proc* 2008; 969: 839.

26. Wei H, Cai Y, Li H, et al. Sambot: a self-assembly modular robot for swarm robot. In: *2010 IEEE international conference on robotics and automation*, Anchorage, Alaska, USA, 3–8 May 2010. IEEE.

27. Wei H, Cui Y, Li H, et al. Kinematics and the implementation of a modular caterpillar robot in trapezoidal wave locomotion. *Int J Adv Robot Syst* 2013; 10(8): 304–403.

28. Davey J, Kwon N, and Yim M. Emulating self-reconfigurable robots-design of the SMORES system. In: *IEEE/RSJ international conference on intelligent robots & systems*, Vilamoura-Algarve, Portugal, 7–12 October 2012. IEEE.

29. Casal A and Yim MH. Self-reconfiguration planning for a class of modular robots. *SPIE Proceed* 1999; 3839: 246–257.

30. Butler Z, Fitch R, and Rus D. Experiments in distributed locomotion with a unit-compressible modular robot. In: *IEEE/RSJ international conference on intelligent robots & systems*, Lausanne, Switzerland, 30 September–4 October 2002. IEEE.

31. Stoy K, Shen WM, and Will PM. A simple approach to the control of locomotion in self-reconfigurable robots. *Robot Autonom Syst* 2003; 44(3): 191–199.

32. Liu C and Yim M. Configuration recognition with distributed information for modular robots. In: Amato N., Hager G, et al. (eds). *Robotics Research*, 2019;11(10). Cham: Springer.

33. Liu C, Whitzer M, and Yim M. A distributed reconfiguration planning algorithm for modular robots. *IEEE Robot Automat Lett* 2019; PP(99): 1–1.

34. Ma L, Sun H, and Song J. Fractional-order adaptive integral hierarchical sliding mode control method for high-speed linear motion of spherical robot. *IEEE Access* 2020; 8: 66243–66256.

35. Hou WF. Distributed, dynamic, and autonomous reconfiguration planning for chain-type self-reconfigurable robots. In: *IEEE international conference on robotics & automation*, Pasadena, CA, 19–23 May. IEEE.

36. Dutta A, Dasgupta P, and Nelson C. Distributed configuration formation with modular robots using (sub)graph isomorphism-based approach. *Auton Robots* 2018; 43: 837–857.