Robotica (2005) volume 23, pp. 581–594. © 2005 Cambridge University Press
doi:10.1017/S0263574704001092 Printed in the United Kingdom

# Optimal topological transformation of underwater modular self-reconfigurable robots

X. S. Xu*, T. Ge and J. M. Zhu

*Underwater Engineering Institute, Shanghai Jiao Tong University, Shanghai, 200030 (P. R. China)*

(Received in Final Form: October 3, 2004)

**SUMMARY**

Underwater Modular Self-Reconfigurable (UMSR) robots, made up of many identical modules, can adjust their configurations to multiple underwater environments or tasks. They can be used in many complex underwater occasions where the ROVs/AUVs can't work well. However, their reconfiguration is difficult because this has to involve many connection adding and removing operations which are difficult to be executed in the underwater environment. To reduce the times of these operations, we propose the theory of Topological Transformation (TT), which includes some definitions in TT, the basic approach to TT, and the Genetic Algorithm (GA) based solution for the optimal TT.

KEYWORDS: Topological transformation; Connectivity planning; Optimal reconfiguration; Modular self-reconfigurable robots.

## I. INTRODUCTION

UMSR robots, composed of many identical modules, can adjust their configurations to multiple underwater environments or tasks by series of connections (or disconnections) and relative movements between modules. They can be used in many complex occasions such as obstructive offshore explorations, inconvenient underwater salvages, intricate oceanic structure inspections etc., where the conventional ROVs/AUVs[1,2] can't work well. Compared with conventional underwater explorations tools,[3] the UMSR robots have the following advantages:

- Self-maintenance. Due to unit-modularity, the redundant modules can replace the malfunctioning ones, so the local failure will not affect the systemic working.
- Multi-locomotion mode. UMSR robots comprise many independent modules each of which can moderate its buoyancy so that the robots can go down to the water floor and crawl on it, or suspend in water and swim or propel forward, or stick upward to a underwater structure and trace on it, and so on.
- High-adaptability. UMSR robots can initiatively gather some environmental information such as terrain, geology and water current, and so on, and then decide which configuration best for the environments. For instance, they can become four-footed robots to creep on the bumpy floor, and then reconfigure into cricoid shape to roll on flat surface.

* Corresponding author: xxsdoc@sjtu.edu.cn

The UMSR robotics can be viewed as the extension of land Modular Self-Reconfigurable (MSR) robotics, despite the fact that their reconfiguration is more difficult than that of the general MSR robots. To detail the features of UMSR robots, we firstly review the information of MSR robots.

### *I.1. Review of MSR robots*

MSR robots have the ability of changing configuration automatically, adapting themselves to suit changing environments or tasks, and their underlying design philosophy is to build complicated systems from a varying number of modules. The basic premise behind MSR robots is twofold:[4,5]

- Systems are made up of any number of identical interconnected modules (Modularity).
- The modules have the ability to move to connect and disconnect from one another, thereby changing the overall shape, or configuration, of the robots (Reconfigurability).

Most of the current robots can be grouped into three categories: lattice MSR robots, chain MSR robots, and prismatic MSR robots.

In lattice robots, a module needs a substrate of other modules to move over in order to get to a new location.[6] The modules can be seen to occupy discrete positions in a lattice, and are constrained to move over the exterior surface or inside cavities. No internal motions are possible when the modules are closely packed. These robots can closely approximate solids and 3D surfaces, and well suited to creating static and deformable solid structures. Lattice robots are the most popular category, despite the facet that the hardware implementation of a module is typically complex, requiring more actuating and connecting parts than in chain robots. Examples of 2D and 3D Lattice robots are presented in MTRAN[7] and Proteo.[8,9]

The notable feature of Chain robots apart from the other categories is that the individual modules can independently move throughout the robots. In this category, the module design is such that, in order for a module to have the reachability necessary to move and connect to a physically disparate location on the robot, it must be carried to the goal location by a group, or chain, of other modules. The mechanical design and implementation of chain robots are generally simpler than in the other categories, which also translates into lower cost. Considering that groups of modules

must be moved at a time, torque limits can become an important performance factor for the actuator selection. Polypod[4] and CONRO[10,11] can be sorted into this category.

The modules of Prismatic MSR robots use primistic degrees of freedom to move and attach and detach from others. These robots achieve module relocation by retracting a module into the interior of the module set and then extruding another module at the desired goal location on surface, or vice versa. Overall reconfiguration can be achieved by a repeated number of these operations. The hardware complexity falls between that of lattice and chain robots. The typical examples of this category are TeleCube[12,13] and Crystalline.[14,15]

Except for the three types of MSR robots, there are other specific MSR robots. For instance, I-Cubes are made up of two basic elements: independently controlled, actuated, 3 DOF links, and passive connection cubes.[16,17] CEBOT, or Cellular Robots, consisting of a number of heterogeneous robots, can connect and disconnect from one another to create different compound robots.[18,19]

### *I.2. UMSR robots*

UMSR robots adopt the concept of modularity and reconfigurability of the MSR robots, and extend it to the underwater domain. The target of UMSR robots is to develop novel underwater electomechanic systems which can initiatively collect underwater environmental informations, and decide the best configuration for the underwater environment or task, and then reconfigure to it. However, the electomechanic designs of UMSR robotic modules are much more difficult than the general MSR robots, which mainly come from the design of waterproof connecting parts and many additional integrated parts such as integrated buoyancy modulators and environmental sensors.

UMSR-1 is our first example of UMSR robotics. Considering the module design difficulties of UMSR robots, UMSR-1 adopts the chain style configuration which needs less actuating and connecting parts compared with the lattice or prismatic style configuration.

In our designing scheme, every module of the UMSR-1 robot is made up of 2 rotational submodules and a connecting submodule. A rotational submodule can provide two opposite terminal connecting facets and a rotational DOF. The hardware in this submodule mainly involves rotation and connection actuators, a buoyancy modulator, a controlling chipset, etc. The function of the rotational submodule is to offer the rotational DOF for relative module movements. As shown in the Fig. 1, the rotational submodule prototype has been developed under the underwater rotational connection and buoyancy modulation tests.

A connecting submodule has 6 connecting facets, and its hardware mainly includes many connection actuators and a central processor, etc. Its main function is to offers multi-directional connecting facets. For the present, its prototype is being developed for further tests.

Corresponding with the current submodule prototype development, a virtual model (shown in Fig. 2) of UMSR-1 module is used to simulate the real reconfiguration problems, which has 2 rotational submodules on two sides and a connecting submodule in the middle, and provides six connecting facets and 2 perpendicular rotational Degree Of Freedoms

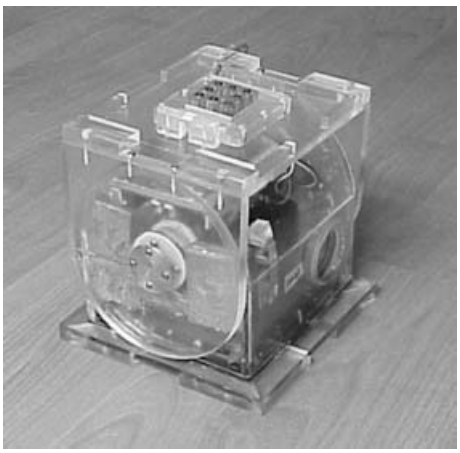


Fig. 1. A rotational submodule.

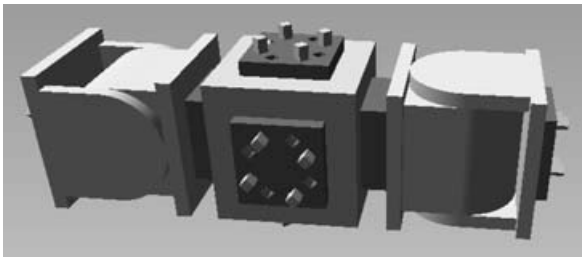


Fig. 2. A virtual module of the UMSR-1 robot.

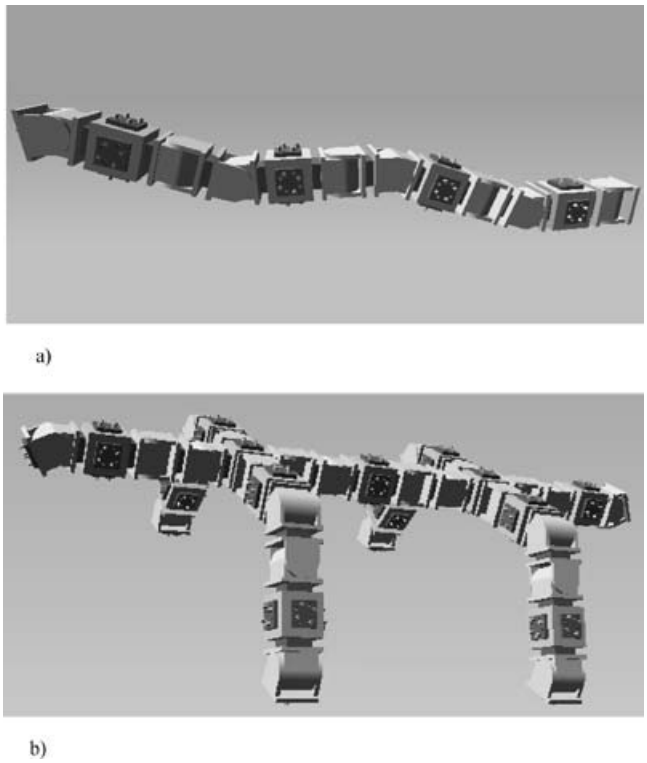



Fig. 3. Two UMSR robots: (a) A 5-module snake-like robot; (b) A 13-module four-footed robot.

(DOFs). Employing the virtual modules as the basic building units, many polymorphic robots can be constructed. Fig. 3(a) shows a 5-module snake-like robot, and Fig. 3(b) shows a 13-module four-footed robot. The number of modules is chosen

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

based on the need of the possible application, generally about from 5 to 50. Too many modules may cause the difficulties of underwater communication and coordination, and also result in the high cost of system manufacture.

### *I.3. The TT problem of UMSR robots*

Similar to the other chain-style MSR robots, the reconfiguration of UMSR robots is accomplished by series of connections (or disconnections) and relative movements between modules. The TT is referred to the changing of configuration topology (which will defined in section 2.2) by a sequence of connection adding or removing operations between modules of UMSR robots. And the aim of the TT research is to find out the sequence of these operations. Of course, the connection adding operations may actually need the motion planning to compute the connection paths between modules so as to accomplish them. But in this paper, we are mainly concerned with the TT problem, i.e. deciding the sequence of connection adding or removing operations in the reconfiguration of UMSR robots.

UMSR robots work in underwater environments, where the connection adding or removing operations are difficult to be executed because the power and communication line in the connector must be waterproof during these operations. So in the TT, the times of these operations must be reduced minimally. Moreover the times reduction of these operations means the reduction of module movements, and thereby results in the reduction of energy consumption which is critical for the robotic underwater survival.

Due to the application difference between MSR robots and UMSR robots, the existing research on the reconfiguration planning of MSR robots is mainly focused on the module motions, i.e. the reconfiguring strategy of module movements,[7,15–17] and the target of the optimal reconfiguration is to reduce the relative movement between modules.[6] Though a HSD method[4] proposed by Casal can be applied to PolyBot, a chain MSR robot, for connectivity planning, it don't consider reducing the times of connection adding or removing operations since PolyBot aims to the land application in which these operations are easy to be fulfilled.

In this paper, we endeavor to solve the optimal TT problem, i.e. minimally to reduce the times of connection adding or removing operations in the UMSR robotic reconfiguration. The remainder of the paper is organized as follows: In Section 2, we propose a basic approach to TT. In this section, we introduce the graph representation of UMSR robotic configuration, give some definitions in TT, and then detail the basic approach to TT based on these definitions. In Section 3, we focus on the optimal TT, and propose a GA based solution for this optimization problem. In Section 4, we summarize the work presented in the paper.

## II. BASIC APPROACH TO TT

To illustrate our TT approach, this section firstly introduces the graph representation of the UMSR robotic configuration, which converts a modular robotic configuration to a 2-dimensional graph. By the graph representation, this section defines some concepts in TT. And then based on the definitions, the section gives the basic approach to TT, and induces the principle of the optimal TT.

### *II.1. Graph representation of UMSR robotic configuration*

Before addressing the graph representation, we introduce the numbers and types of connecting facets. As shown in Fig. 4, the connecting facets in a module are numbered from 1 to 6, which are sorted into two classes: Class *I*, including facets numbered from 1 to 2, and Class *II* including facets numbered from 3 to 6, according to the asymmetry of the connecting facets. Obviously, the connection on a class-*I* facet can provide a rotational DOF, while the connection on a class-*II* facet is rigid. The difference of connecting facets results in the two different kinds of connections between two modules: the linear connection (Fig. 5(a)) and the cross connection (Fig. 5(b)). All UMSR robotic configurations can be constructed with these two kinds of connections.

For a given UMSR robotic configuration, its graph representation should reflect all informations of modules and their connections for the convenience of the robotic description. So in the graph representation as shown in Fig. 6 a encircled number represents a module in UMSR robots, a line between two circled numbers corresponds a connection between two modules, and the two numbers at the two ends of a line adjacent with the encircled numbers denote the connecting facets of the two modules. By the graph representation, it is easier to obtain the connectivity of UMSR robotic modules.

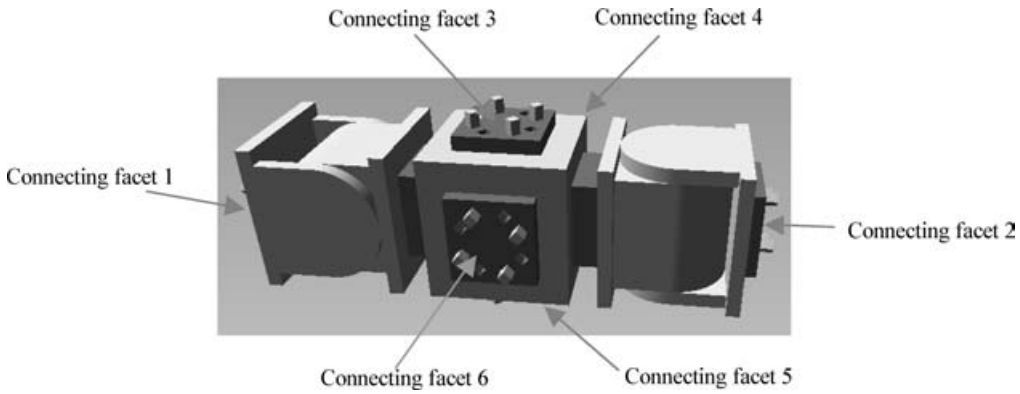
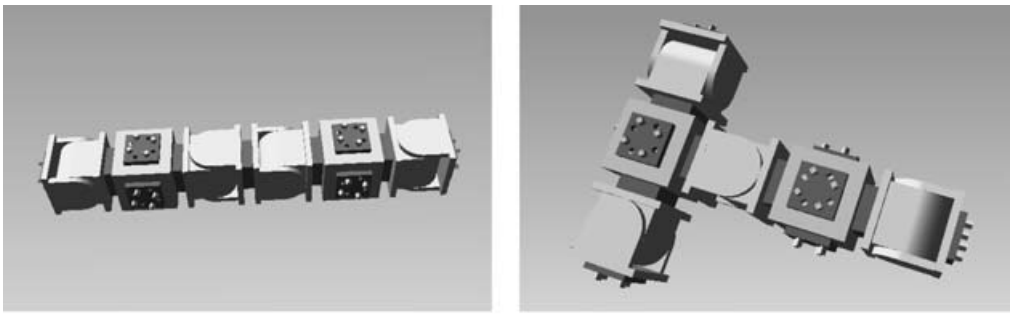



Fig. 4. The six connecting facets of a UMSR module.

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press



a) b)

Fig. 5. Two kinds of connections: (a) A linear connection, formed by two class-*I* connecting facets; (b) A cross connection, formed by a class-*I* connecting facet and a class-*II* connecting facet.

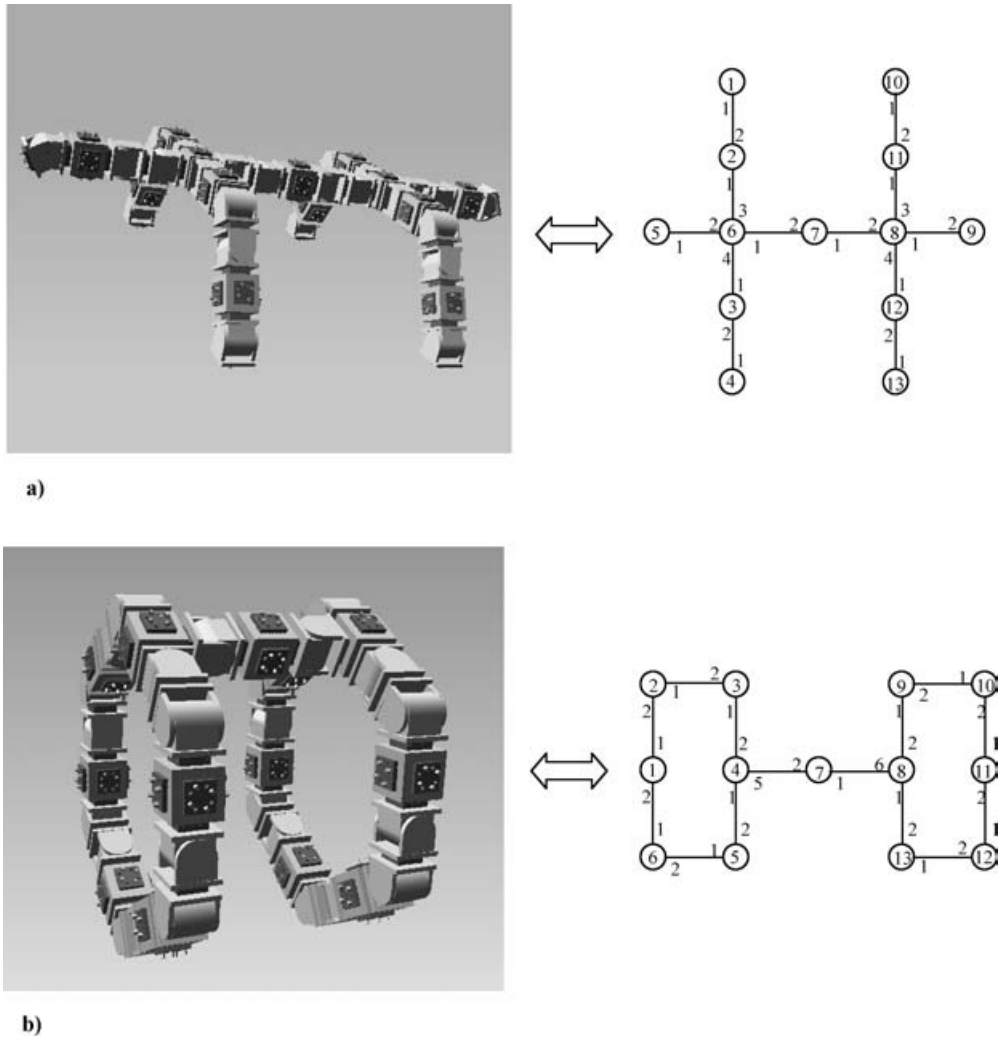
*II.2. Definitions in TT*

The *connection* can be seen as the attachment of two connecting facets of different modules, written as ($a<f_{ab}>$, $b<f_{ba}>$), where $a$ and $b$ are the module numbers, and $f_{ab}$ and $f_{ba}$ are the connecting facet numbers of the respective modules. For example, in Fig. 6 (a), the connection (1<1>,2<2>) is referred to the attachment of the connecting facet 1 of the 1st module and the connecting facet 2 of the 2nd module.

The *connection path* is referred to the set of sequent connections between two modules. For example, in Fig. 6(a), the connection path between the 1st and 7th modules can be written as {(1<1>,2<2>), (2<1>,6<3>), (6<1>,7<2>)}.

The connection set of is *connective* if there is a connection path between any two modules in the set.

The *size* of a connection set is referred to the number of connections in the set.



a) b)

Fig. 6. Two 13-module UMSR robotic configurations and their graph representation: (a) A 13-module four-footed robotic configuration and its graph representation; (b) A 13-module two-wheeled robotic configuration and its graph representation.

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

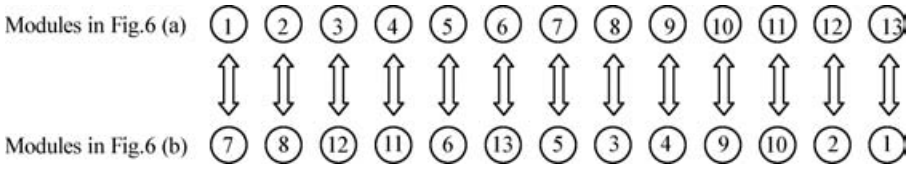



Fig. 7. The MM expressed by the value (7,8,12,11,6,13,5,3,4,9,10,2,1).

The *Configuration Topology* (*CT*) here is defined as the connective set of all connections in a configuration.

**Example 1**: The CT of the four-foot configuration $F^{(1)}$ in Fig. 6(a) can be expressed as follow:

$T^{(1)} = \{$(1<1>,2<2>), (2<1>,6<3>), (3<2>,4<1>), (3<1>, 6<4>), (5<1>,6<2>), (6<1>,7<2>), (7<1>,8<2>), (8<1>, 9<2>), (8<3>,11<1>), (8<4>,12<1>), (10<1>,11<2>), (12<2>, 13<1>)$\}$

Similarly, the CT of the two-wheel configuration $F^{(2)}$ in Fig. 6(b) can be expressed as follow:

$T^{(2)} = \{$(1<1>,2<2>), (1<2>,6<1>), (2<1>,3<2>), (3<1>, 4<2>), (4<1>,5<2>), (4<5>,7<2>), (5<1>,6<2>), (7<1>, 8<6>), (8<2>,9<1>), (8<1>,13<2>), (9<2>,10<1>), (10<2>, 11<1>), (11<2>,12<1>), (12<2>,13<1>)$\}$

The size of $T^{(1)}$ is 12, and the size of $T^{(2)}$ is 14.

The *configuration* can be viewed as an entity of modules and a CT on these modules.

To compare two configurations, the *Module Matching* (MM) is introduced. For the two configurations $F^{(a)}$ and $F^{(b)}$ with the same number of modules, the MM between $F^{(a)}$ and $F^{(b)}$ is referred to the bi-projective matching between the modules of $F^{(a)}$ and the modules of $F^{(b)}$, expressed by the value $(v_1,v_2,v_3,\ldots,v_n)$, where $n$ is the number of modules, $v_i$ represents the matching between the $i$th module in $F^{(a)}$ and the $v_i^{\text{th}}$ module in $F^{(b)}$. For example, in the MM value (7,8,12,11,6,13,5,3,4,9,10,2,1) between $F^{(1)}$ and $F^{(2)}$ in Example 1, the first subvalue '7' means the matching between the 1st module in $F^{(1)}$ and the 7th module in $F^{(2)}$. The meanings of the other MM subvalues are similar to the first, this MM can be also expressed by the Fig. 7.

Supposing there is a MM value $V$ between the configuration $F^{(a)}$ and the configuration $F^{(b)}$, the CT of $F^{(a)}$is $T^{(a)}$, the CT of $F^{(b)}$ is $T^{(b)}$, by $V$:

- The connection $c^{(a)}$ in $T^{(a)}$ is *equivalent* to the connection $c^{(b)}$ in $T^{(b)}$ if the connecting facets of the matching modules are in the same class.
- The connection set $C^{(a)}$ in $F^{(a)}$ is *equivalent* to the connection set $C^{(b)}$ in $F^{(b)}$ if every connection in $C^{(a)}$ can find an equivalent connection in $C^{(b)}$, and vice versa.
- The *Maximal Common Subset* (MCS) between $T^{(a)}$ and $T^{(b)}$ is referred to the set of all these connections that belong to $T^{(a)}$ and have equivalent counterparts in $T^{(b)}$.
- Supposing the connection set $C^{(a1)} \subset T^{(a)}$, $C^{(b1)} \subset T^{(b)}$, the *addition* of $C^{(a1)}$ and $C^{(b1)}$ is referred to the connection set in which the connections belong to $C^{(a1)}$ or have equivalent counterparts in $C^{(b1)}$ , expressed by $C^{(a1)} + C^{(b1)}$ ; the *subtraction* of $C^{(a1)}$ and $C^{(b1)}$ is referred to the connection set in which the connections belong to in $C^{(a1)}$ and have no equivalent counterpart in $C^{(b1)}$, expressed by $C^{(a1)} - C^{(b1)}$. Especially, the addition and subtraction of two connection sets in a configuration can be seen to operate by a self-projective MM value.
- Supposing $C^{(a,b)}$ is the MCS between $T^{(a)}$ and $T^{(b)}$, the *TT* from $T^{(a)}$ to $T^{(b)}$ is a process of removing the connections that aren't in $C^{(a,b)}$, but in $T_n^{(a)}$, and adding the connections that aren't in $C^{(a,b)}$, but have equivalent counterparts in $C^{(b)}$.

**Example 2**: in Example 1, by the MM value (7,8,12,11, 6,13,5,3,4,9,10,2,1) between $F^{(1)}$ and $F^{(2)}$:

- (12<2>,13<1>) in $T^{(1)}$ is equivalent to (1<1>,2<2>) in $T^{(2)}$, because the connecting facets of the 12th module of $F^{(1)}$ and the 2st module of $F^{(2)}$ are class-$I$ facets, and the connecting facets of the 13th module in $F^{(1)}$ and the 1st module in $F^{(2)}$ are class-$I$ facets too.
- The connection set $C^{(1)} = \{$(3<2>,4<1>), (8<1>,9<2>), (10<1>,11<2>), (12<2>,13<1>)$\} \subset T^{(1)}$, $C^{(2)} = \{$(11<2>, 12<1>), (3<1>,4<2>), (9<2>,10<1>), (1<1>,2<2>)$\} \subset T^{(2)}$, and every connection in $C^{(1)}$ have an equivalent connection in $C^{(2)}$ and vice versa, thus $C^{(1)}$ is equivalent to $C^{(2)}$.
- Moreover, except the connections in $C^{(1)}$, there is no more connection in $T^{(1)}$ that has equivalent connection in $T^{(2)}$, thus $C^{(1)}$ is the MCS between $T^{(1)}$ and $T^{(2)}$.
- The TT from $T^{(1)}$ to $T^{(2)}$ is a process of removing connections in ($T^{(1)}$–$C^{(1)}$) and adding connections that have equivalent counterparts in ($T^{(2)}$–$C^{(2)}$).

*II.3. Basic Approach to TT*

Based on the above definitions, we propose the following three-step basic approach to the TT in the UMSR robotic reconfiguration:

(i) Select a MM value for the TT.
(ii) Find out the connection set to be removed, and the equivalent connection set to be added for the TT.
(iii) Search for the proper sequence of connection adding and removing operations to accomplish the TT.

Supposing there are two configurations $F^{(a)}$ and $F^{(b)}$, the CT of $F^{(a)}$ is $T^{(a)}$, the CT of $F^{(b)}$ is $T^{(b)}$, the three-step approach to the TT from $T^{(a)}$ to $T^{(b)}$ can detail as following:

In the first step, we should select a MM value $V$ between $F^{(a)}$ and $F^{(b)}$ for this TT.

In the second step, we should find out the connection set $C_r$ to be removed, and the equivalent connection set $C_a$ to be added in the TT. To compute $C_r$ and $C_a$, we should find the MCS $C^{(a,b)}$ between $T^{(a)}$ and $T^{(b)}$.

The connection set to be removed can be computed by $C_r = T^{(a)} - C^{(a,b)}$ , and the equivalent connection set to be

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

added can be computed by $C_a = T^{(b)} - C^{(a,b)}$. Notice that $C_r$ and $C^{(a,b)}$ are in $F^{(a)}$, but $C_a$ is in $F^{(b)}$.

In the last step, we should find out the proper sequence of the connection adding or removing operations from $C_r$ and $C_a$. This step is more complicated than the above two steps. In this paper we propose a Tree-searching algorithm for searching the proper sequence, which processes as follows:

(i) Initialize the connection set $C_{rt}$ to be removed, the connection set $C_{at}$ to be added, and the CT $T_t^{(a)}$ of $F^{(a)}$ : $C_{rt} = C_r$, $C_{at} = C_a$, $T_t^{(a)} = T^{(a)}$.
(ii) Set up a TT tree, attach the information of $\{C_{rt}, C_{at}, T_t^{(a)}\}$ to the root of the tree, and set the root as the current branch.
(iii) If there is no removable connection in $C_{rt}$, find out the addible connection set $C_{ac}$ in $C_{at}$. Update $C_{at}, T_t^{(a)}$ : $C_{at} = C_{at} - C_{ac}$, and $T_t^{(a)} = T_t^{(a)} + C_{ac}$, add a new branch with the information of $\{C_{rt}, C_{at}, T_t^{(a)}\}$ to the current branch, set the current branch as the ancestor of the new branch, set the new branch as the current branch.
(iv) If $T_t^{(a)}$ is equivalent to $T^{(b)}$, go to (viii).
(v) If there is no removable connection in $C_{rt}$ , go to 7).
(vi) Update $C_{rt}, C_{at}$ and $T_t^{(a)}$ from the current branch. Find out the removable connection set $C_{rc}$ in $C_{rt}$, for every connection $c_r$ in $C_{rc}$:
  vi.a Remove $c_r$ from $C_{rc}$, $C_{rt}$ and $T_t^{(a)}$, i.e. $C_{rc} = C_{rc} - \{c_r\}$, $C_{rt} = C_{rt} - \{c_r\}$, $T_t^{(a)} = T_t^{(a)} - \{c_r\}$;
  vi.b Find the addible connection set $C_{ac}$ in $C_{at}$, update $C_{at}$ and $T_t^{(a)}$: $C_{at} = C_{at} - C_{ac}$, $T_t^{(a)} = T_t^{(a)} + C_{ac}$;
  vi.c Add a new branch with the information of $\{C_{rt}, C_{at}, T_t^{(a)}\}$ to the current branch, and set the current branch as the ancestor of this branch, then set this new branch as the current branch;
  vi.d Analyze whether there is a removable connection in $C_{rt}$:
  6.4.1) If yes, go to (vi);
  6.4.2) Else if $T_t^{(a)}$ of the current branch is equivalent to the goal CT $T^{(b)}$, it succeeds, go to (viii);
  vi.e Set the ancestor of the current branch as the current branch;
(vii) No feasible sequence for this TT problem, step out of the program.
(viii) Record the path from the current branch to the root of the tree, figure out the sequence of connection adding or removing operations from the path, go out of the program.

In the above Tree-searching algorithm, the removable connection is referred to this without which the configuration keeps connective. But for the addible connection, the following conditions must be satisfied:

- The addible connection must have enough remaining connecting facets for the modules of the connection. But the connection between two class-*II* facets is prohibited, because it will result in rotational collision between adjacent modules.
- The addible connection must have an equivalent counterpart in the CT of the goal configuration.
- The addible connection must have 6 or more DOFs for space movement between two modules of the connection.

The above Tree-searching algorithm searches the sequence solution by depth, and $T_t^{(a)}$ in the last branch will be equivalent to the goal CT $T^{(b)}$. However some MM values are infeasible, because the operation sequence can't be searched out by them. For the infeasible MM values, a lot of additional connection adding and removing operations must be added in, which will complicate the TT problem. So if we meet the infeasible MM value, we must choose another MM value.

**Example 3**: To accomplish the TT from $T^{(1)}$ to $T^{(2)}$ in Example 1.

In the first step, select the MM value (7,8,12,11,6,13, 5,3,4,9,10,2,1).

In the second step, we should find out the connection set $C_r$ to be removed, and the equivalent connection set $C_a$ to be added in the TT. To compute $C_r$ and $C_a$, we should find out the MCS $C^{(1,2)}$ between $T^{(1)}$ and $T^{(2)}$.

$C^{(1,2)} = \{$(3<2>,4<1>), (8<1>,9<2>), (10<1>,11<2>), (12<2>,13<1>)$\}$

The connection set $C_r$ to be removed can be expressed by:

$C_r = T^{(1)} - C^{(1,2)} = \{$(1<1>,2<2>), (2<1>,6<3>), (3<1>, 6<4>), (5<1>,6<2>), (6<1>,7<2>), (7<1>,8<2>), (8<3>, 11<1>), (8<4>,12<1>)$\}$

The equivalent connection set $C_a$ to be added can be expressed by:

$C_a = T^{(2)} - C^{(1,2)} = \{$(1<2>,6<1>), (2<1>,3<2>), (4<1>, 5<2>), (4<5>,7<2>), (5<1>,6<2>), (7<1>,8<6>), (8<2>, 9<1>), (8<1>,13<2>), (10<2>,11<1>), (12<2>,13<1>)$\}$

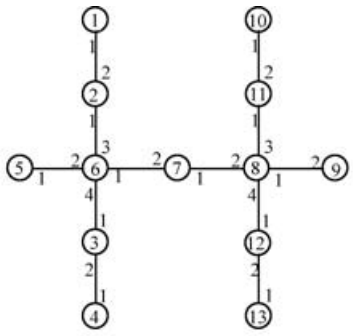
In the third step, searching out the sequence of the connection adding or removing operations from $C_r$ and $C_a$. Employing the Tree-searching algorithm, the following sequence can be searching out:

1) Adding (5<2>,13<2>) and (1<2>,9<5>)
2) Removing (8<4>,12<1>)
3) Removing (2<1>,6<3>), adding (2<1>,10<2>)
4) Removing (8<3>,11<1>), adding (4<2>,11<1>)
5) Removing (7<1>,8<2>), adding (8<2>,12<1>)
6) Removing (5<1>,6<2>), adding (7<1>,9<1>)
7) Removing (3<1>,6<4>), adding (3<1>,6<2>)
8) Removing (1<1>,2<2>), adding (1<1>,2<6>)
9) Removing (6<1>,7<2>), adding (5<1>,7<2>) and (2<2>, 6<1>)

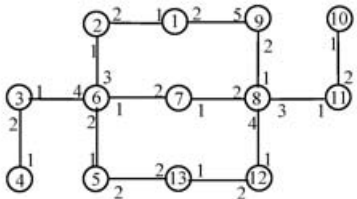
According to this sequence, the TT from $T^{(1)}$ to $T^{(2)}$ is shown in Fig. 8.

From the above basic approach, we can see: for a feasible MM value between $F^{(a)}$ and $F^{(b)}$, $T^{(a)}$ is the CT of $F^{(a)}$, $T^{(b)}$ is the CT of $F^{(b)}$, $C^{(a,b)}$ is the MCS between $T^{(a)}$ and $T^{(b)}$, the result of TT from $T^{(a)}$ to $T^{(b)}$ is to remove connections in $(T^{(a)} - C^{(a,b)})$ from $T^{(a)}$, and add some connections which have equivalent counterpart in $(T^{(b)} - C^{(a,b)})$ to $T^{(a)}$, so **the times of connection adding or removing operations is**

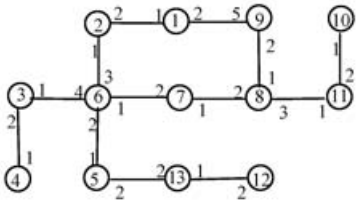
https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press



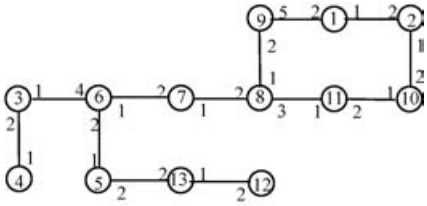
(1) Initial confiugration



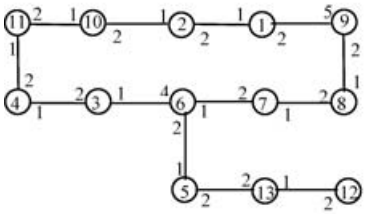
(2) Adding (5<2>,13<2>)
and (1<2>,9<5>)



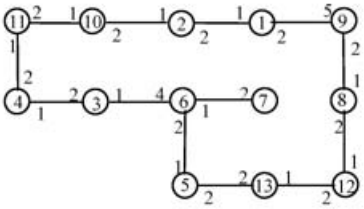
(3) Removing (8<4>,12<1>)



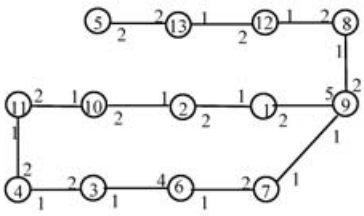
(4) Removing (2<1>,6<3>),
adding (2<1>,10<2>)



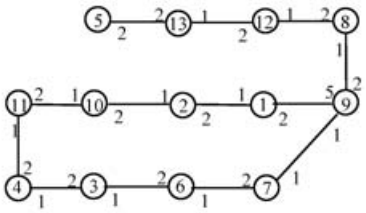
(5) Removing (8<3>,11<1>),
adding (4<2>,11<1>)



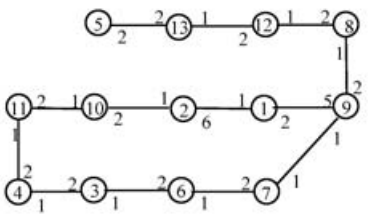
(6) Removing (7<1>,8<2>),
adding (8<2>,12<1>)



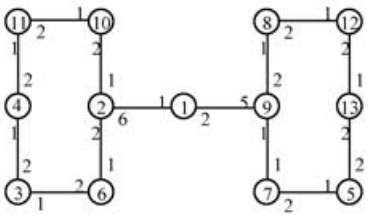
(7) Removing (5<1>,6<2>),
adding (7<1>,9<1>)



(8) Removing (3<1>,6<4>),
adding (3<1>,6<2>)



(9) Removing (1<1>,2<2>),
adding (1<1>,2<6>)



(10) Removing (6<1>,7<2>),
adding (5<1>,7<2>)and (2<2>,6<1>)

Fig. 8. The TT from $T^{(1)}$ to $T^{(2)}$ by the MM value (7,8,12,11,6,13,5,3,4,9,10,2,1).

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

**equal to the subtraction of the total size of $T^{(a)}$ and $T^{(b)}$ by the double size of $T^{(a,b)}$, expressed by**

$$\textbf{Times} = S^{(a)} + S^{(b)} - 2*S^{(a,b)}.$$

Where $S^{(a)}$ is the size of $T^{(a)}$, $S^{(b)}$ is the size of $T^{(b)}$, and $S^{(a,b)}$ is the size of $C^{(a,b)}$.

In Example 3, $S^{(a)} = 12$, $S^{(b)} = 14$, $S^{(a,b)} = 4$, hence the times of connection adding or removing operations is 18.

From the above expression, we can see that if we want to reduce the times of connection adding or removing operations, we should increase the size of the MCS, which is decided by the adopted MM value. So, for the optimal TT, we should try to find the optimal feasible MM value, i.e. the MM value by which the maximal size of the MCS can be obtained.

## III. GA BASED SOLUTION TO THE OPTIMAL TT

From the above section, we have known that the key of the optimal TT is to find the optimal MM value. Since both the MM values and the search space are discrete in nature, combinatorial optimization techniques should be applied.

Exhaustive search techniques can find the exact optimal matching values, but the search space is so large that implementation of such an algorithm becomes infeasible. For a TT between two UMSR robots with $n$ modules, there are $n!$ MM values according to the permutation and combination theory. Supposing it will take about $P$ computations for a MM value to judge its feasibility and get the size of the MCS, there will need about $P \cdot n!$ computations for two $n$-module robots. If $n = 20$, $P = 10^4$, there will needs about $2.4*10^{22}$ computations to get the exact optimal solution. Obviously, exhaustive search techniques aren't fit for this discrete optimal problem.

Due to the huge times of computations, we have to adopt some probabilistic search techniques such as the Genetic Algorithm (GA) and the Simulated Annealing (SA) methods to solve this problem. In this paper, we use the GA method.

### *III.1. GA based searching for the optimal MM value*

The GA is a probabilistic search method based on the principle of evolution and hereditary of nature systems.[20,21] In such an algorithm, a population of individuals for each generation is maintained. The individual is implemented with suitable data structure and is evaluated by a 'fitness function' to give a measure of its 'fitness'. A new population is formed through selecting the more suitable individuals. In this procedure, an individual with larger 'fitness' value is more likely to be selected for the new generation. Some members of the new generation undergo transformations by the 'genetic operators' to form new solutions. Probability rules are applied to determine the execution of the operators. Two types of genetic operators are considered in general: Crossover type and mutation type. The crossover operation reorganizes data segments in several individuals to form new individuals. The mutation operator makes small changes in a single individual. After some number of generations, the individuals will converge to the optimal or nearly optimal solution.

**III.1.1. Coding scheme.** It is reported that the computational efficiency and stability of GAs rely primarily on the data representation method, i.e. the coding scheme. Classical GAs usually use fixed-length binary string as a chromosome (the data structure) for its individuals and the genetic operators: the binary mutation and the binary crossover. Such a coding scheme when adopted for robot configuration designs becomes inconvenient because the binary string representation is not sufficient to reflect the nature of the design parameters. So in our GA, there are some alterations including employing the problem-specific data structure to represent the chromosomes and modifying the 'genetic' operators appropriate for the given data structure. Since the MM value $(v_1, v_2, \ldots, v_n)$ conceptually defines the data structure of the solution space, we adopt the MM value as the chromosome and defined the MM related genetic operators. The main objective behind such an implementation is to move the GA closer to the solution space.

**III.1.2. Fitness evaluation.** The objective of optimization is to find out the optimal or near-optimal feasible MM value by which we can get the maximal or near-maximal size of the MCS. According to this, the fitness evaluation for a MM value needs two-step computation: First, analyzing the MCS and get the size of it, And then judging whether the MM value is feasible or not, i.e. confirming there is a sequence of connection adding or removing operations for the TT using the Tree-searching algorithm in subsection 3.3. If the MM value is feasible, i.e. the sequence exists, its fitness value adopts the size of the MCS; or else, the fitness value is zero. The fitness function can be expressed by:

$$F(V) = \begin{cases} S^{(a,b)} & \text{if } V \text{ is feasible} \\ 0 & \text{if } V \text{ is unfeasible} \end{cases}$$

Where $V$ is a MM value, $S^{(a,b)}$ is the size of the MCS between two CTs by V.

Obviously, a MM value with bigger fitness value will have more chance of getting the next generation.

**III.1.3. Generation scheme.** The generation scheme is proposed to automatically generate the initial population of MM values. The scheme is executed by a random selector that produces the give number of population from arbitrary permutations of the all module numbers. For example, we can randomly select a MM value (6,5,7,9,8,4,3,2,1,10,13,11,12) as an individual from a permutation of module numbers of a 13-module robot. Repeating the selection the given number of times, we can obtain the given number of population.

**III.1.4. Crossover operator.** The crossover operator applies to two MM values. As shown in Fig. 9, a cut-off line is randomly chosen in the latter half of the MM values. The subvalues in the two matching values after the cut-off line are called pre-swapped subvalues. However, this operator only swaps the *pre-swapped* subvalues that have no equal subvalues in the counterpart pre-swapped subvalues. The subvalues before the cut-off line will not swapped, but if they are equal to some swapped subvalues, they should be changed into the counterpart swapped subvalues.

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

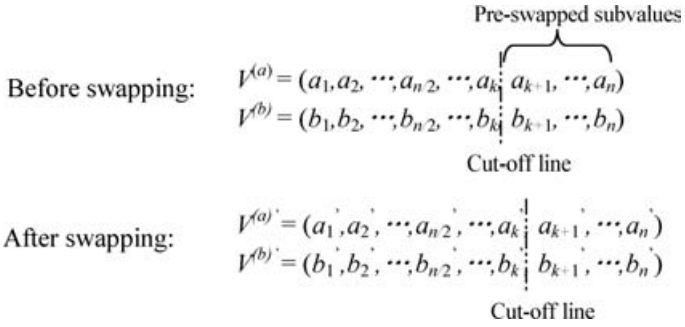



Fig. 9. The crossover operation between $V^{(a)}$ and $V^{(b)}$.

The cross-over operation in Fig. 9 can be processed as the following steps:

(i) After the cut-off line, swapping $a_i$ and $b_i$, if $a_i \neq b_j$ and $b_i \neq a_j$, where $k \geq n/2$, $k+1 \leq i \leq n$ and $k+1 \leq j \leq n$;
(ii) Before the cut-off line, $a_i = b_j$, if $a_i = a_j$, where $1 \leq i \leq k$ and $k+1 \leq j \leq n$.

As shown in Fig. 10, there are three examples of the cross-over operation. In Fig. 10 (a), there is a crossover operation between (2,4,5,7,1,13,9,6,12,10,3,11,8) and (11,3,4,8,2,13, 12,10,1,9,5,6,7). In this example, there is no equal subvalue after the cut-off line in two MM values, so all the pre-swapped subvalues participate in swapping. And if the subvalues before the cut-off line have the equal subvalues after the cut-off line, change their value according to the cross-over operation step 2).

In Fig. 10 (b), there is an equal subvalue '3' in two MM values after the cut-off line, so only two subvalues after the cut-off line participate in swapping for the cross-over operation. The changing of subvalues before the cut-off line is similar with the past example in (a).

The reason why the equal subvalues after the cut-off line cannot participate in swapping is that if these subvalues go to swapping, it will lead to some repeated subvalues in the MM values. As shown in Fig. 10 (c), if the equal subvalues '3' execute swapping, they will reach the improper results.

**III.1.5. Mutation operator.** The mutation operator applies to a single module matching value, which comprises the swapping suboperator and the circulating suboperator. The swapping suboperator is accomplished by swapping any two selected subvalues, while the circulating suboperator is accomplished by circulating all subvalues left or right at any times.

For example, for a MM value $V^{(a)} = (2,4,5,7,1,13,9,6, 12,10,3,11,8)$, after swapping subvalue '9' with '12', and circulating all subvalues right 2 times, it becomes $V^{(a)'} = (11,8,2,4,5,7,1,13,12,6,9,10,3)$.

**III.1.6. Implementation of the GA.** The proposed GA is successfully implemented with C++ codes in the Microsoft Development Environment. As Fig. 11 shows, in the overall algorithm, the input parameters include the number of modules $n$, the population size $n_p$, the number of destination generation $n_g$, the probability of the crossover operator $p_c$, and the probability of the mutation operator $p_m$. The first step is to randomly generate $n_p$ MM values for the initial

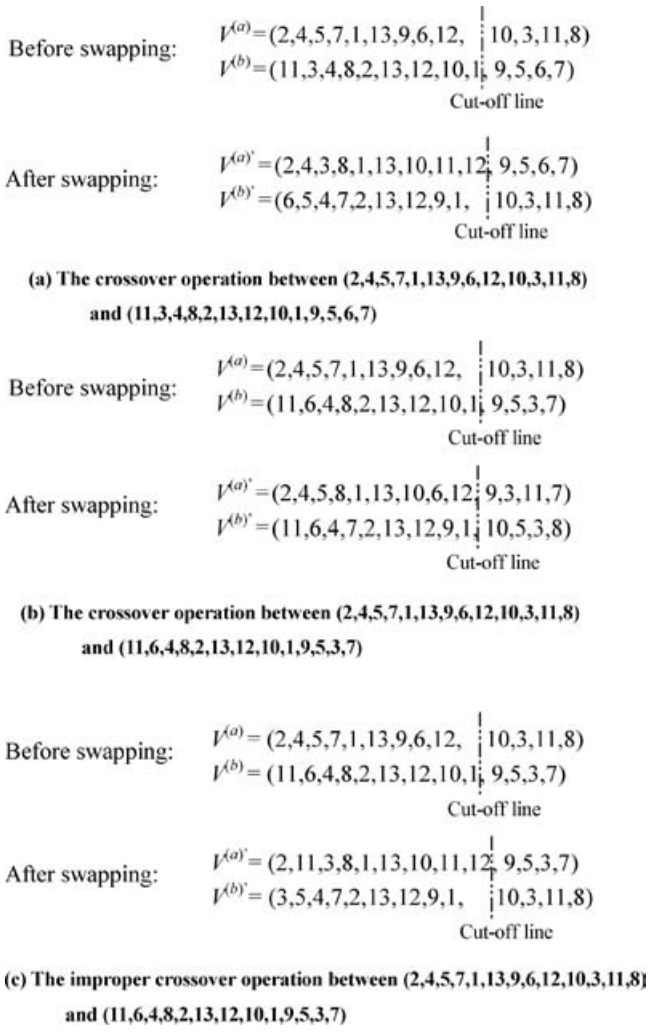



Fig. 10. Three examples of crossover operation: (a) The cross-over operation between (2,4,5,7,1,13,9,6,12,10,3,11,8) and (11,3,4, 8,2,13,12,10,1,9,5,6,7); (b) The crossover operation between (2,4,5,7,1,13,9,6,12,10,3,11,8) and (11,6,4,8,2,13,12,10,1,9,5,3,7); (c) The improper crossover operation between (2,4,5,7,1,13,9,6, 12,10,3,11,8) and (11,6,4,8,2,13,12,10,1,9,5,3,7).

generation by using the generation scheme. It is followed by the fitness evaluation procedure to evaluate the fitness value for every MM value. After fitness evaluation procedure, a new generation of MM values is produced through reproduction, crossover, and mutation operations. The entire process will be repeated until the predetermined number of generation $n_g$ is reached. In the final generation, we choose the MM value with the maximal fitness value as the optimal or near optimal MM value.

**III.1.7. Constrained optimal MM value searching in TT.** The above MM value optimization is based on the consideration of the general TT problem, but in some cases there are some special things that cannot be ignored. For instance, there is a module with some sensors or cameras that must match another specific module of goal configuration so as to keep relative position of these devices unchanged after the TT. This case can be simply viewed as that there is a constraint in the TT that two modules of the respective configuations must always keep matching.

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

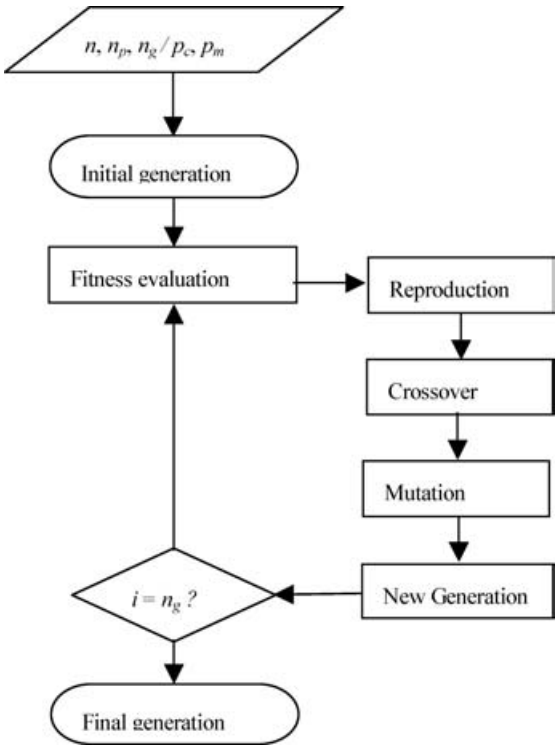



Fig. 11. The implementation of GA based MM value optimization.

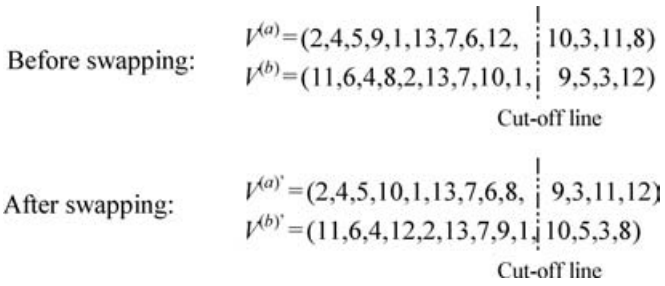



Fig. 12. A constrained crossover operation between (2,4,5,9,1,13,7,6,12,10,3,11,8) and (11,6,4,8,2,13,7,10,1,9,5,3,12).

For the constrained TT, the GA algorithm can show great advantages. The only difference of the constrained MM value searching from the general case is that the constrained subvalues always keep unchanged in the searching process, or rather, except for the constrained subvalues, the GA-based searching operations on the other subvalues in the constrained case are the same with these in the general case.

For example, supposing the 7th subvalue of a MM value must take '7' after the TT. As Fig. 12 shows, the subvalue '7' keeps unchanged after swapping in the crossover operation.

And the mutation operation on a MM value can accomplished as the following:

For $V^{(a)} = (2,4,5,9,1,13,7,6,12,10,3,11,8)$, after swapping between subvalue '9' and subvalue '12', and circulating all subvalues right 2 times, it becomes $V^{(a)'} = (11,8,2,4,5,12,7,1,13,6,9,10,3)$.

**III.1.8. Time complexity of GA based searching algorithm.** In the above GA based searching algorithm, the computations are mainly spent on the three steps: the evaluation on all individuals, the individual selection by

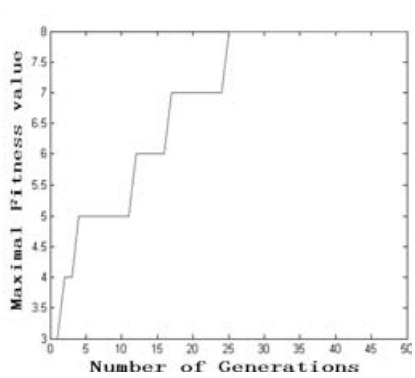



Fig. 13. The result of general TT based on GA optimization.

individual fitness and the genetic operations on the selected individuals. Supposing it needs $P$ computations for a MM value to compute the fitness of all individuals, the evaluation on all individuals will need $n_p \cdot n_g \cdot P$ computations. The individual selection will need about $n_p^2 \cdot n_g$ computations. The genetic operations will need about $n^2 \cdot n_p \cdot n_g$ computations. So the above GA based searching algorithm will need about $(n_p \cdot n_g \cdot P + n_p^2 \cdot n_g + n^2 \cdot n_p \cdot n_g)$ computations. If n = 20, $n_p = 200$, $n_g = 50$, $P = 10^4$, the algorithm will need about $1.06*10^8$ computations. Compared with the exhaustive search which needs $n! \cdot P$ computations, the GA based searching algorithm need much fewer computations.

*III.2. GA based solution to the optimal TT problem*

By taking the above GA searching, we can find out the optimal MM value for a TT problem. And then by this optimal MM value, we can employ the basic approach in section II.3. to find the sequence of connection removing or adding operations for the optimal TT. Owning to the maximal size of the MCS, the times of the connection removing or adding operations in the TT will be reduced to the minimum.

In order to demonstrate the effectiveness of the proposed GA based optimal TT, two examples are present in this subsection. In the first example (Example 4), we try to use the proposed GA based solution to solve the general optimal TT form $T^{(1)}$ to $T^{(2)}$ in Example 1. After the GA based MM value searching, we can find out the near-optimal MM value. Then by this value, we can find out the sequence of connection adding or removing operations. From this example, we can see the GA based solution can effectively reduce the times of connection operations in the TT. In the second example (Example 5), a constraint is added to the same TT problem, we can also find a desirable near-optimal result by the GA based solution. In these two examples, the parameters of GA are taken as the number of modules $n = 13$, the size of population $n_p = 200$, and the number of goal generation $n_g = 50$.

**Example 4**: To employ the GA based solution to accomplish the TT from $T^{(1)}$ to $T^{(2)}$ in Example 1 without constraint.

Firstly, we should find out the optimal MM value with the proposed GA based searching. The parameters of genetic operators are taken as $p_c = 0.6$, $p_m = 0.1$. The result of genetic algorithm is shown in Fig. 13. After 25 times of

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

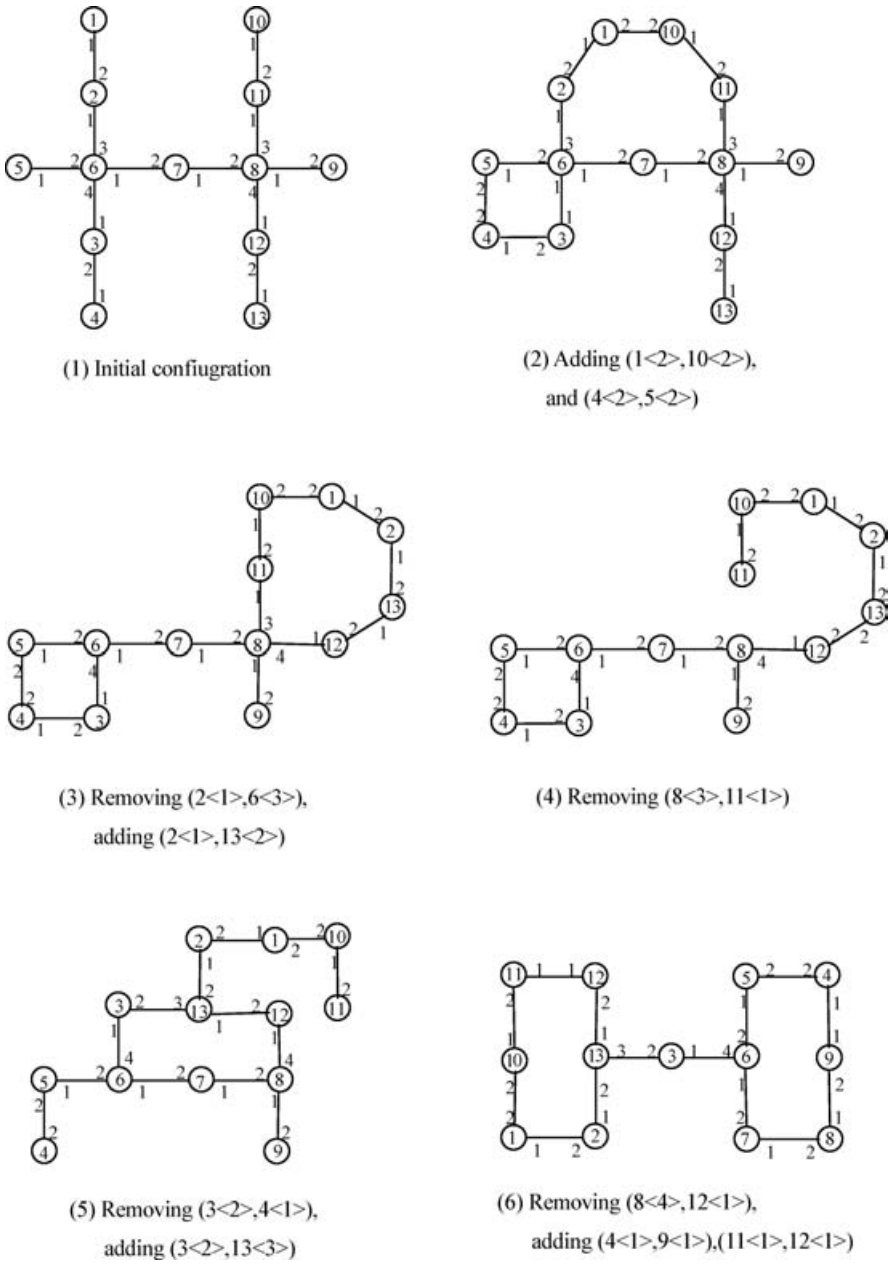



Fig. 14. The optimal TT from $T^{(1)}$ to $T^{(2)}$ by the MM value (6,5,7,10,9,8,13,12,11,1,2,3,4).

generation selection, the program reaches the near-optimal MM value $V_{opt} = (6,5,7,10,9,8,13,12,11,1,2,3,4)$.

By this MM value, the MCS between $T^{(1)}$ and $T^{(2)}$ can be expressed by:

$C^{(1,2)} = \{$(1<1>,2<2>), (3<1>,6<4>), (5<1>,6<2>), (6<1>,7<2>), (7<1>,8<2>), (8<1>,9<2>), (10<1>,11<2>), (12<2>,13<1>)$\}$

By this MM value, employing the basic approach in Section 2, we can find out the sequence of connection adding or removing operations:

(i) Adding (1<2>,10<2>) and (4<2>,5<2>)
(ii) Removing (2<1>,6<3>), adding (2<1>,13<2>)
(iii) Removing (8<3>,11<1>)
(iv) Removing (3<2>,4<1>), adding (3<2>,13<3>)
(v) Removing (8<4>,12<1>), adding (4<1>,9<1>), (11<1>, 12<1>)

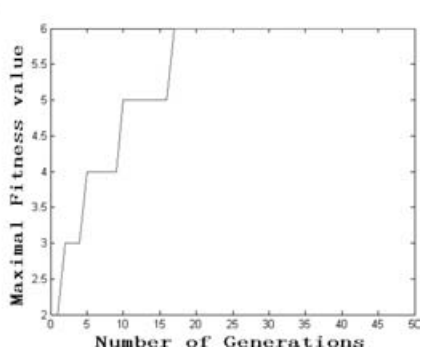



Fig. 15. The result of a constrained TT based on GA optimization.

The whole TT process is shown in Fig. 14, which needs only 10 times of connection adding or removing operations.

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

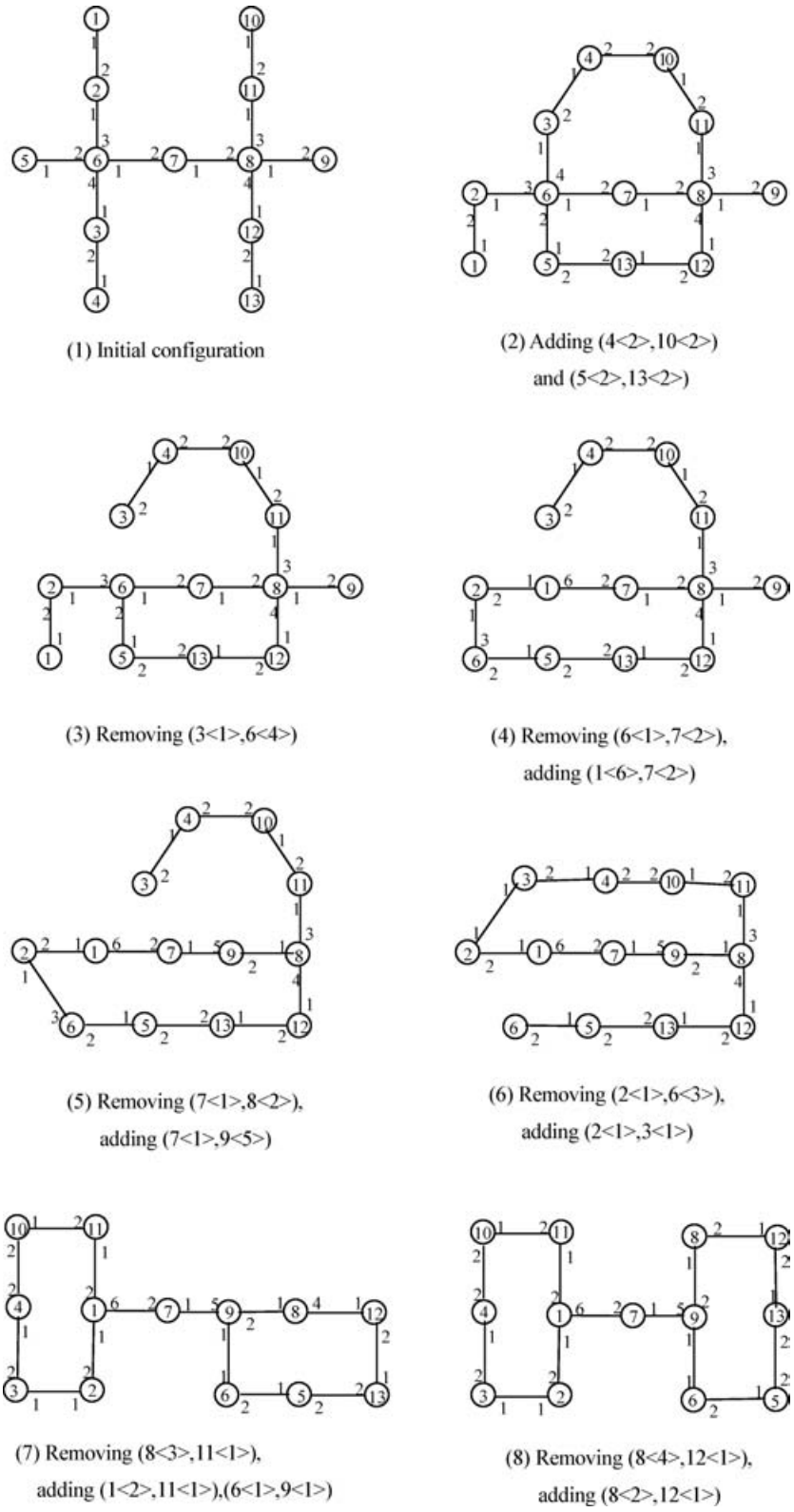



Fig. 16. The optimal constrained TT from $T^{(1)}$ to $T^{(2)}$ by the MM value (4,3,2,1,10,9,7,13,8,6,5,12,11).

Compared with the TT in Fig. 8 which needs 18 times of these operations, 8 times of operations is saved owing to the bigger size of MCS by this near-optimal MM value.

**Example 5**: To employ the GA based solution to accomplish the TT from $T^{(1)}$ to $T^{(2)}$ in Example 1 with a constraint.

If there is a constraint in the topological transformation, the result may be different from the general case. In this example, the 7th module in $F^{(1)}$ must match the 7th module in $F^{(2)}$ since some devices such as cameras must be mounted on the same relative mid place.

Firstly, we can employ the proposed GA based searching to get the optimal MM value in which the seventh subvalue must take '7' and by which the TT from $T^{(1)}$ to $T^{(2)}$ needs the minimal times of connection operations. In this example, the

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

genetic operator is taken as $p_c = 0.6$, $p_m = 0.2$. The result of GA is shown in Fig. 15, after 17 times of generation selection, the program have gotten the near-optimal MM value $V_{opt} = (4,3,2,1,10,9,7,13,8,6,5,12,11)$.

By this MM value, the common MCS between $T^{(1)}$ and $T^{(2)}$ can be expressed by:

$C^{(1,2)} = \{(1<1>,2<2>), (3<2>,4<1>), (5<1>,6<2>), (8<1>,9<2>), (10<1>,11<2>), (12<2>,13<1>)\}$

By this MM value, employing the basic approach in Section 2, we can find out the sequence of connection adding or removing operations:

(i) Adding (4<2>,10<2>) and (5<2>,13<2>)
(ii) Removing (3<1>,6<4>)
(iii) Removing (6<1>,7<2>), adding(1<6>,7<2>)
(iv) Removing (7<1>,8<2>), adding(7<1>,9<5>)
(v) Removing (2<1>,6<3>), adding (2<1>,3<1>)
(vi) Removing (8<3>,11<1>), adding (1<2>,11<1>) and (6<1>,9<1>)
(vii) Removing (8<4>,12<1>), adding (8<2>,12<1>)

The whole constrained TT process is shown in Fig. 16, which needs 14 times of connection adding or removing operations. Compared with the TT in Fig. 8 which needs 18 times of these operations, 4 times of these operations is avoided. From this example, we can see that the times of connection adding or removing operations in the constrained TT also can be reduced with the proposed GA based solution.

## IV. CONCLUSIONS

Different from the conventional underwater exploring tools, USMR robots have the characteristics of modularity and reconfigurablity. The modularity endows UMSR robots with good maintainability for the redundant modules in robotic system can replace the broken-down ones. The reconfigurability provides them with good adaptability to multiple environments or tasks. However, the reconfiguration of UMSR robots has to involve many connections adding and removing operations which are difficult to be executed in the underwater environments for the waterproof reason. So the timing of these operations should be reduced to the minimum. To reduce the times, we propose the TT theory which includes the basic approach to the TT, and the GA based solution for the optimal TT problem.

In the basic approach to the TT, we presents some definitions in TT, bring up the basic approach to TT based on the definitions, and then sums up the basic principle of the optimal TT; the optimal TT should be accomplished by the feasible optimal MM value with the maximal size of the MCS.

To find the optimal MM value, we adopt the GA based optimization, present the detailed implementation scheme of the optimization, and then come up with the GA based solution for the optimal TT problem. After that, we give two examples to illustrate the GA based solution. From those two examples, we can see the GA based solution can not only effectively reduce the times of connection adding or removing operations in the general TT, but also solve the constrained TT problem.

## Acknowledgments

This research is supported by Chinese National Nature Scientific project (#60104001), Chinese National "863" High Technology project (#2001AA616090) and Shanghai Qimingxing project (#01QF14028).

## References

1. R. Wernli and J. Jaeger, "ROV Technology Update From an International Perspective", *OCEANS*, *Proceedings* (Sept, 1984) **Vol. 16**, pp. 634–645.
2. R. Damus, J. Manley, S. Desset, etc., "Design of an Inspection Class Autonomous Underwater Vehicle", *Proceeding of Oceans' 02 MTS/IEEE* (Oct., 2002) **Vol. 1**, pp. 180–185.
3. J. Akizono, T. Tanaka, K. Nakagawa, etc., "Seabottom roughness measurement by aquatic walking robot", *OCEANS' 97. MTS/IEEE Conference Proceedings* (6–9 Oct., 1997) **Vol. 2**, pp. 1395–1398.
4. Aranzazu Casal, "Reconfiguration planning for modular self-reconfigurable robots", *Ph.D. thesis* (Stanford University, USA, 2002).
5. Mark Yim, "Locomotion with a unit-module reconfigurable robot", *Ph.D. thesis* (Stanford University, USA, 1994).
6. A. Pamecha, I. Ebert-Uphoff and G. S. Chirikjian, "Useful metrics for modular robot motion planning", *Robotics and Automation, IEEE Transactions* **13**, 531–545 (August, 1997).
7. E. Yoshida, S. Murata, A. Kamimura, etc., "Reconfiguration planning for a self-assembling modular robot", *Assembly and Task Planning, 2001, Proceedings of the IEEE International Symposium* (28–29 May, 2001) pp. 276–281.
8. H. Bojinov, A. Casal and T. Hogg, "Multiagent control of self-reconfigurable robots", *MultiAgent Systems, 2000. Proceedings. Fourth International Conference* (10–12 July 2000) pp. 143–150.
9. H. Bojinov, A. Casal and T. Hogg, "Emergent structures in modular self-reconfigurable robots", *Robotics and Automation, 2000. Proceedings. ICRA' 00. IEEE International Conference* (24–28 April, 2000) **Vol. 2**, pp. 1734–1741.
10. Shen, W. M., B. Salemi and P. Will, "Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots", *Robotics and Automation, IEEE Transactions* **18**, 700–712 (Oct., 2002).
11. A. Castano and P. Will, "Representing and discovering the configuration of Conro robots", *Robotics and Automation, Proceedings 2001 ICRA, IEEE International Conference* (2001) **Vol. 4** pp. 3503–3509.
12. J. W. Suh, S. B. Homans and M. Yim, "Telecubes: mechanical design of a module for self-reconfigurable robotics", *Robotics and Automation, ICRA' 02. IEEE International Conference* (11–15 May, 2002) **Vol. 4**, pp. 4095–4101.
13. S. Vassilvitskii, M. Yim and J. Suh, "A complete, local and parallel reconfiguration algorithm for cube style modular robots", *Robotics and Automation, ICRA' 02. IEEE International Conference* (11–15 May, 2002) **Vol. 1**, pp. 117–122.
14. D. Rus and M. Vona, "A physical implementation of the self-reconfiguring crystalline robot", *Robotics and Automation, 2000. Proceedings. ICRA' 00. IEEE International Conference* (24–28 April, 2000) **Vol. 2**, pp. 1726–1733.
15. D. Rus and M. Vona, "Self-reconfiguration planning with compressible unit modules", *Robotics and Automation*, 1999. *Proceedings. 1999 IEEE International Conference* (10–15 May, 1999), **Vol. 4**, pp. 2513–2520.
16. C. Unsal and P. K. Khosla, "A multi-layered planner for self-reconfiguration of a uniform group of I-Cube modules", *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference* (29 Oct.–3 Nov., 2001) **Vol. 1**, pp. 598–605.

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press

17. K. C. Prevas, C. Unsal, M. O. Efe, etc., "A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system", *Robotics and Automation, 2002. Proceedings. ICRA' 02. IEEE International Conference* (11–15 May, 2002) **Vol. 1**, pp. 787–792.
18. T. Fukuda, S. Nakagawa, Y. Kawauchi, etc., "Structure decision method for self organising robots based on cell structures-CEBOT", *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference* (14–19 May, 1989) **Vol. 2**, pp. 695–700.
19. H. C. An, T. Fukuda, F. Arai, etc., "Hierarchical control architecture for Cellular Robotic System-simulations and experiments", *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference* (21–27 May, 1995) **Vol. 1**, pp. 1191–1196.
20. N. Chaiyaratana and A. M. S. Zalzala, "Recent developments in evolutionary and genetic algorithms: theory and applications", *Genetic Algorithms In Engineering Systems: Innovations And Applications, 1997. GALESIA 97. Second International Conference* (Conf. Publ. No. 446), (2–4 Sept., 1997) pp. 270–277.
21. W. K. Chung, Han Jeongheon, Y. Youm, etc., "Task based design of modular robot manipulator using efficient genetic algorithm", *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference* (20–25 April, 1997) **Vol. 1**, pp. 507–512.

https://doi.org/10.1017/S0263574704001092 Published online by Cambridge University Press