Robotica (2014) volume 32, pp. 723–741. © Cambridge University Press 2013
doi:10.1017/S0263574713001136

Curve shortening-inspired self-reconfiguration of heterogenous hexagonal-shaped modules toward a straight chain

Yizhou Miao, Gangfeng Yan and Zhiyun Lin*

State Key Laboratory of Industrial Control Technology, College of Electrical Engineering, Zhejiang University, 38 Zheda Road, Hangzhou, 310027 P. R. China

(Accepted October 29, 2013. First published online: December 2, 2013)

SUMMARY

This study deals with a self-reconfiguration problem of hexagonal-shaped modules from an arbitrary initial configuration to a straight chain. Modules are modeled as the same-sized rigid bodies. Two categories of modules with different functionalities are used. One category comprises two powerful modules, which are expected to play the role of terminal modules in a goal configuration. The other category comprises several ordinary modules, which are expected to fill in the middle portion in a goal configuration. A distributed control strategy, inspired by the idea of curve shortening, is developed for each module to act cooperatively to attain a goal configuration. It is verified that under the proposed strategy, modules eventually converge to a straight chain.

KEYWORDS: Modular robots; Motion planning; Control of robotic systems; Swarm robotics; Multi-robot systems.

1. Introduction

A modular system may contain any finite number of autonomous modules. The modules have similar appearances and computation abilities, and obey similar locomotion rules. Modules' connecting/disconnecting to/from one another results in the reconfiguration of modular configuration. Generally, the shape of a module looks symmetric, e.g., cubic,[1] hexagonal,[2] rhombic dodecahedral,[3] etc. Self-reconfiguration provides versatility and scalability to a modular system. Thus, it is more adaptive in unknown and complicated environments compared with traditional fixed-morphology robots. Potential applications include self-repairing systems,[4] surgical systems,[5] etc.

Researchers in this field mainly focus on the designing of motion planning strategies and mechanical fabrications. Two different research groups (Chirikjian[6] and Murata *et al.*[7]) simultaneously introduced hexagonal-shaped modular systems. Later, researchers started to study the reconfiguration of modules. Reconfiguration from a simple connected initial configuration to another simple connected final configuration was realized based on a simulated annealing algorithm.[8] Walter *et al.*[9] provided a distributed algorithm for hexagonal modules to attain a simple connected configuration from a chain configuration. Moreover, the reconfiguration from a straight chain configuration to any intersecting chain configurations were addressed.[10] Preliminary geometric properties of a modular system were investigated by Ghrist and Peterson[11] and Zhang and Dai,[12] which provide fresh ideas to view modular systems. Related works with respect to modules' motion planning can be found in refs. [13–19]. To date, several entities of modular systems have been developed.[20–25] These modular systems are based on different modular shapes (hexagonal or cubic), different textures (rigid or compressible), and different reconfiguration methods (attachment or detachment).

It is expected that a modular system can re-shape to arbitrary configurations. Among diverse configurations, a straight chain is one of the basic sub-configurations, since quite a lot of complicated

* Corresponding author. E-mail: linz@zju.edu.cn

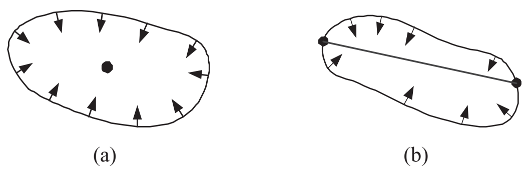


Fig. 1. Curve shortening. (a) Shortened toward a point; (b) shortened toward a line segment with fixed vertices.

configurations can be divided into several straight chains. Thus, in this study we focus on developing distributed strategies for modules to reconfigure toward a straight chain configuration. Suppose there are N modules, among which two are of powerful functionalities such that they are capable of detecting the entire configuration, while the other modules are with limited abilities. According to different functionalities, the two powerful modules are required to occupy terminal positions in a straight chain and the others to occupy middle positions. A novel strategy which is inspired by the idea of *curve shortening* is presented, by which the modular configuration converges to a straight chain in finite time in the absence of centralized control and full knowledge of entire configuration. The contributions lie in the following aspects: (i) A curve shortening-inspired distributed control strategy is developed for the task of achieving a chain configuration, which is considered to be fundamental in autonomous modular systems. (ii) A decomposition idea is proposed to solve the self-reconfiguration problem by transforming a global task into local objectives for different roles of modules. (iii) Rigorous analysis is provided to show the finite-time convergence of the algorithm.

The organization of this study is given as follows: Section 2 contains several preliminary materials and the problem formulation. Sections 3 and 4 are the main parts that provide distributed strategies as well as correctness validation under two different situations respectively. Simulations and discussions are addressed in Section 5. Section 6 concludes the work and lists the future work.

2. Preliminaries and Problem Formulation

2.1. *Hexagonal lattice and lattice distance*

In this study, every module is modeled as a rigid regular-hexagonal entity with the same size. Modules are assumed to operate in a perfect hexagonal lattice consisting of regular hexagonal cells, in which each cell is of the same size and shape as a module. The coordinate system in a hexagonal lattice was first introduced by Chirikjian[6] with its origin at the center of a cell, its x-axis perpendicular to an edge of lattice cell, and its y-axis being determined by rotating the x-axis $\pi/3$ radians counterclockwise. The distance in a hexagonal lattice is called *lattice distance* and the lattice distance between two cells, $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$, is calculated according to the following formula:

$$LD(c_1, c_2) = \begin{cases} \max(|\Delta x|, |\Delta y|) & \text{if } \Delta x \cdot \Delta y < 0 \\ |\Delta x| + |\Delta y| & \text{otherwise} \end{cases}, \tag{1}$$

where $\Delta x = x_1 - x_2$ and $\Delta y = y_1 - y_2$. The lattice distance indicates the least number of steps needed for one module to move from c_1 to c_2. Hence, components of a cell coordinate are integers.

2.2. *Curve shortening and polytope*

Given a smooth closed curve in a planar Cartesian coordinate system, one can shorten it by letting every point in it move toward the direction of the corresponding curvature vector filed.[26,27] Generally, such a curve would shorten until it reaches a point (Fig. 1(a)). But if two points in the curve are fixed, the curve would eventually shorten toward a line segment with the fixed two points acting as vertices (Fig. 1(b)).

Given a set S, its *convex hull* is the smallest convex set containing S. Moreover, suppose S contains a finite number of points: $\{p_1, \ p_2, \ldots, p_N\}$. Its convex hull is also called a *polytope* (Fig. 2(a)), and is denoted as $P = Co\{p_1, \ p_2, \ldots, p_N\}$. In a similar way, suppose there are a finite number of modules in a hexagonal lattice. Let us define a *polytope* in a hexagonal lattice as the convex hull containing all the modules where each edge of a polytope should cross the center of a cell and be

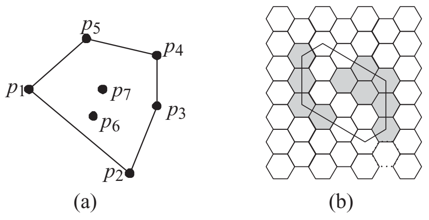


Fig. 2. Polytopes (a) in a Cartesian coordinate system; (b) in a lattice coordinate system.

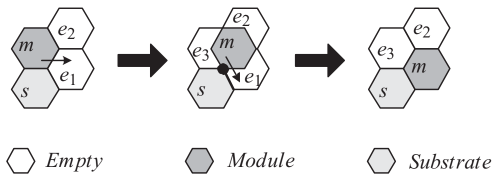


Fig. 3. Before (left), and after (right) locomotion.

perpendicular to an edge of the cell (Fig. 2(b)). For a given configuration, the polytope is formed by drawing three pairs of parallel lines (namely, one pair of lines parallel to the x-axis, the second pair of lines parallel to the y-axis, and the third pair of lines parallel to the y-axis rotated by $\pi/3$ radians counterclockwise) such that the configuration is enclosed and each line contains at least one module. So it is clear that a polytope in a hexagonal lattice has at most six vertices, and is at most a hexagon.

2.3. *Problem formulation*

We consider the following problem in the paper.

Problem 1. *Design a local information-based distributed reconfiguration strategy for each module such that the modular system reconfigures from arbitrary initial configuration to a straight chain configuration.*

The concepts of *straight chain configuration* and *collinearity* are referred to Walter *et al.*[10] Denote

$$\kappa(c_i, c_j) := \min\left\{|x_i - x_j|, |y_i - y_j|, |(x_i + y_i) - (x_j + y_j)|\right\}$$

as the degree of collinearity between any two distinct cells, $c_i(x_i, y_i)$ and $c_j(x_j, y_j)$. It is intuitive that $\kappa(c_i, c_j) = 0$ if and only if c_i and c_j are collinear. Besides, under the same lattice distance, the smaller the value of κ, the less steps needed for two modules locating on these two cells to achieve a collinear distribution.

Throughout this study, we denote a *round* as a constant time period within which a module would accomplish certain behaviors to update to an adjacent empty cell if possible. Concurrent motions are allowed in a round, and asynchronous mechanism is simulated for preventing potential collisions from happening. Physical restrictions are taken into consideration for a rigid module with the locomotion mechanism being designed as follows.

Definition 1. *A module first slides along the common edge of one of its adjacent modules called* substrate *that remains static in the same round, and next slides into an empty cell along another edge of the substrate.*

An illustration of the locomotion mechanism is given in Fig. 3. It infers that in order to perform a one-step updating, a module needs one adjacent module and two adjacent empty cells such that the three cells are contiguously spaced. A candidate empty cell e_1 for a module m to move into is called a *feasible cell*, which satisfies the following properties:

- e_1 is adjacent to module m;
- a common adjacent cell of e_1 and module m is empty, while the other is not.

A module with at least one feasible cell is said to be *active*, and a module without feasible cells is said to be *inactive*. In addition, a module is said to be *isolated* if all of its adjacent cells are empty. An isolated module cannot locomote according to Definition 1.

Two different kinds of modules with different functionalities are used in our study to fulfill the goal configuration cooperatively. Let S_1 and S_2 represent the set of two kinds of modules, respectively. Specifically, S_1 contains two powerful modules, and S_2 contains the other $N-2$ modules (called *ordinary modules*). Without loss of generality, let m_1, m_2 denote powerful modules, and $m_3, \ldots, m_N$ denote ordinary modules (actually labels are not necessary information for modules).

Quite commonly, a configuration of a group of modules is represented in a global reference frame. If global coordinates of the modules are all available and all-to-all communications are allowed, it is then trivial to attain a desired configuration. However, it is usually impossible for distributed modules to access all global coordinates in a global reference frame and to have all-to-all communications. Thus, we concentrate on developing distributed control strategies via local interaction with neighboring modules in absence of a global reference frame and all-to-all communications for this objective. Inspired by the idea of *global-to-local* programming put forward by Yamins and Nagpal,[28] we would like to decompose Problem 1 into the following two subproblems.

Subproblem 1. *Design a distributed reconfiguration strategy for each powerful module such that the two powerful modules reach a collinear distribution with $N-1$ cells lattice distance.*

Subproblem 2. *Design a distributed reconfiguration strategy for each ordinary module such that each ordinary module reaches a location collinear with all of its neighbors and also adjacent to its neighbors.*

The proposition below will show the equivalence of Problem 1 and the union of Subproblems 1 and 2.

Proposition 1. *The straight chain configuration in Problem 1 is achieved with both terminal nodes being the powerful modules if and only if both Subproblems 1 and 2 are achieved.*

Proof. (Necessity) If the current configuration is a straight chain configuration with both terminal nodes being the powerful modules, then it is certain that the two powerful modules are collinearly distributed, and the lattice distance between them equals $N-1$. At the same time, each ordinary module has two adjacent modules, and is collinear with all neighbors.

(Sufficiency) Suppose, in contradiction, that the current configuration is not a straight chain configuration with both terminal nodes being the powerful modules. Therefore, either collinearity is not satisfactory, or there exists a terminal node being an ordinary module, although the configuration is a straight chain. For the former situation, at least one module should not have reach a collinear distribution with its neighbors. For the latter case, either the lattice distance between the two powerful modules does not equal to $N-1$, or there exists an ordinary module (acting as a terminate node) which has only one adjacent neighbor. Both cases contradict the statements in Subproblem 1 and Subproblem 2 respectively. $\square$

At the end of this section, several assumptions are summarized.

(A_1) The two powerful modules are able to obtain the relative positions of each other.

(A_2) All modules are able to detect the relative positions of the nearest modules in each direction.

(A_3) Communication only occurs between adjacent modules. The communication message transmitted by a powerful module contains only information (logic values) to indicate that it has reached the goal in Subproblem 1, and whether it is located on the boundary of its local polytope. The communication message transmitted by an ordinary module contains the following information (also logic values): (1) The status whether it has reached the goal in Subproblem 2 or the equivalent terminating condition. (2) Its understanding about the status of whether a powerful module has reached the goal in Subproblem 1, and whether the powerful module is located on the boundary of its local polytope.

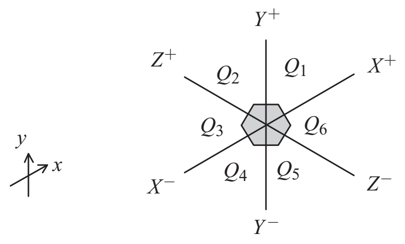


Fig. 4. A module divides the lattice plane into 12 parts.

3. Convergence of Ordinary Modules

This section concentrates on solving Subproblem 2, with the assumption that the two powerful modules initially have been fixed at desired positions, i.e., they are collinearly spaced with $N - 1$ cells lattice distance. The initial positions of ordinary module are randomly distributed without any isolated module. The idea to solve Subproblem 2 can be summarized as follows. Each ordinary module determines its local polytope according to the union of its neighbors' positions and the self-position; after knowing the local polytope, a module is then able to determine whether it is located on the boundary or strictly interior to the local polytope; and design a reconfiguration strategy for each ordinary module to shrink the local polytope till a straight chain.

3.1. Self-identification for modules

An intuitive way to make a polytope shrinking is to let modules which are located on the boundary to their local polytopes select feasible cells interior to their local polytopes to move to. It then follows the question about how can a module, using limited local information, differentiate its role, i.e., whether it is located on the boundary or interior to its local polytope, and in addition, the relationship between one's local polytope and the polytope of the entire configuration. A novel method will be introduced in this section to make modules possible to differentiate self-roles.

Given a module positioned on an arbitrary cell in the lattice, we can divide the lattice plane into six quadrants, denoted as $Q_1\sim Q_6$ by inscribing three lines across the center of the module such that two of them are parallel to x-axis and y-axis respectively and the third by rotating y-axis $\pi/3$ radians counterclockwise. Denote by X^+, X^-, Y^+, Y^-, Z^+, and Z^- the six axes with each separating adjacent quadrants respectively. Regard the six quadrants plus the six axes as 12 distinct directions. The union of these 12 directions together with the given module compose the entire lattice (Fig. 4).

Consider an arbitrary module $m_i(x_i, y_i)$. The origin of m_i's local lattice coordinate frame is set at some cell, and the x-axis and y-axis are parallel with that illustrated in Fig. 4. Let $m_j(x_j, y_j)$ be another module in the system measured in m_i's local frame. Then the relationship between relative distribution of m_i, m_j, and the 12 directions satisfies the following formulas (positions are measured in the same local frame):

$$\begin{aligned}
X^+ &= \{(x_j, y_j) \mid x_j > x_i,\ y_j = y_i\},\\
X^- &= \{(x_j, y_j) \mid x_j < x_i,\ y_j = y_i\},\\
Y^+ &= \{(x_j, y_j) \mid x_j = x_i,\ y_j > y_i\},\\
Y^- &= \{(x_j, y_j) \mid x_j = x_i,\ y_j < y_i\},\\
Z^+ &= \{(x_j, y_j) \mid x_j < x_i,\ x_j + y_j = x_i + y_i\},\\
Z^- &= \{(x_j, y_j) \mid x_j > x_i,\ x_j + y_j = x_i + y_i\},
\end{aligned} \tag{2}$$

$$\begin{aligned}
Q_1 &= \{(x_j, y_j) \mid x_j > x_i,\ y_j > y_i\},\\
Q_2 &= \{(x_j, y_j) \mid x_j < x_i,\ x_j + y_j > x_i + y_i\},\\
Q_3 &= \{(x_j, y_j) \mid x_j < x_i,\ y_j > y_i,\ x_j + y_j < x_i + y_i\},\\
Q_4 &= \{(x_j, y_j) \mid x_j < x_i,\ y_j < y_i\},\\
Q_5 &= \{(x_j, y_j) \mid x_j > x_i,\ x_j + y_j < x_i + y_i\},\\
Q_6 &= \{(x_j, y_j) \mid x_j > x_i,\ y_j < y_i,\ x_j + y_j > x_i + y_i\}.
\end{aligned} \tag{3}$$

In each round, module m_i will send electromagnetic waves to each of the 12 directions. Once meeting with the nearest module in each direction, there is a feedback to m_i, according to which m_i determines whether there exists some modules in that direction. No feedback infers no modules located in that direction.

Remark 1. *Mutual interference may happen due to simultaneous generation of electromagnetic waves from multiple modules. However, as we are only concerned with the direction that has no feedback rather than the direction that has feedback, interference signals can be ignored in the modules' self-role identification.*

Denote

$$N_i(k) = \big\{ j : \mathrm{LD}(m_i, m_j)|_k \le \mathrm{LD}(m_i, m_s)|_k, \\ \forall\ m_j, m_s \text{in the same direction} \big\}$$

as the neighbor set of module m_i in round k. Let $P_i(k)$ represent the local polytope of m_i (corresponding to the configuration comprising m_i and its neighbors). In addition, let $B_i(k)$ and $\mathrm{Int}_i(k)$ represent the boundary and the interior part of $P_i(k)$ respectively. Similarly, let $P(k)$, $B(k)$, and Int (k) represent the same meanings for the entire modular configuration. The convex property of a polytope infers that if a module $m_i \in B_i(k)$, at least five contiguous directions (starting from a quadrant) are without feedbacks. Define a radial angle $\phi_i(k)$, which is the smallest angle covered by the neighbor set of m_i (measured by both starting and ending at axes). As each quadrant covers $\pi/3$ radians in the lattice plane, $\phi_i(k)$ equals to a value that is integer multiple of $\pi/3$. For example, in Fig. 4, if the neighbors of m_i are located in quadrant Q_1 and Q_5, then $\phi_i(k) = \pi$. If the neighbors of m_i are located in quadrant Q_5 and axis X^+, then $\phi_i(k) = 2\pi/3$. If its neighbors are located in quadrants Q_1, Q_4, and axis X^+, then $\phi_i(k) = 4\pi/3$. In conclusion,

$$\begin{cases} m_i \in B_i(k), & \text{if } \phi_i(k) \le \pi \\ m_i \in \mathrm{Int}_i(k), & \text{otherwise} \end{cases}.$$

Moreover,

$$\begin{cases} m_i \in B_i(k) \Rightarrow m_i \in B(k) \\ m_i \in \mathrm{Int}_i(k) \Rightarrow m_i \in \mathrm{Int}(k) \end{cases}.$$

Similar conclusions hold when a module tests its feasible cells.

3.2. An equivalent terminating condition

To achieve the goal configuration, ordinary modules should reach their goals in Subproblem 2 in the same round. Due to the absence of full knowledge about the entire configuration for an ordinary module, it is somewhat difficult for them to reach goals simultaneously, especially when the system contains a large number of modules. This inspires us to look for an equivalent condition to terminate the evolution of an ordinary module. That is, the module becomes collinear with both powerful modules, lies between them, and realizes that powerful modules have reached their subgoals (initially holds in this section). In what follows, we will discuss how an ordinary module can assess the equivalent condition based on limited local information.

Denote by $g_i^c(x_i^c, y_i^c)$ the candidate cell at which the equivalent terminating condition is satisfied for module m_i. Module m_i is able to know g_i^c if (1) it can directly detect relative positions of both powerful modules; (2) it has two adjacent neighboring modules that satisfy the equivalent terminating conditions; and (3) it could eliminate all impossible cases as stated below.

Suppose module $m_i(x_i(k), y_i(k))$ is adjacent to a powerful module (denoted by m_1) in round k. Without loss of generality, suppose the relative position of module m_1 measured by module m_i is $m_1(x_1(k), y_1(k))$=$m_1(x_i(k)$+1, $y_i(k))$. The collinearity property of the equivalent condition implies that

$$\begin{aligned}
&1. y_i^c = y_i(k) \text{ and } x_i^c < x_i(k) + 1, \\
&2. x_i^c = x_i(k) + 1 \text{ and } y_i^c < y_i(k), \\
&3. x_i^c + y_i^c = x_i(k) + 1 + y_i(k) \text{ and } x_i^c > x_i(k) + 1, \\
&4. y_i^c = y_i(k) \text{ and } x_i^c > x_i(k) + 1, \\
&5. x_i^c = x_i(k) + 1 \text{ and } y_i^c > y_i(k), \\
&6. x_i^c + y_i^c = x_i(k) + 1 + y_i(k) \text{ and } x_i^c < x_i(k) + 1.
\end{aligned} \tag{4}$$

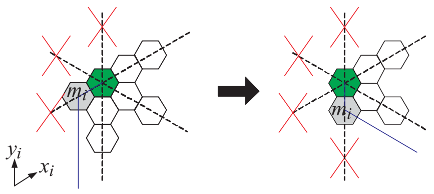


Fig. 5. (Colour online) Finding impossible positions of g_i^c. Red symbol X: impossible cases. Blue lines: an edge of polytope.

Table I. List of symbols used in Algorithm 1.

Symbol	Interpretation	1	0
z_j^i	Relative position of module m_j measured by module m_i	/	/
N_i	Neighbor set of module m_i	/	
F_i	Set for module m_i storing recorded feasible cells	/	/
Achiev_i	State of achievement of local objective for module m_i	Achieved	Not achieved
$f_i(k)$	Feasible cell of module m_i according to the locomotion mechanism	/	/
G_i^c	Set of g_i^c	/	/
Iso_i	State of generating potential isolated modules, provided that module m_i moves	Generated	Not generated
z_i	Position of module m_i	/	/
$\mathbf{N}(0,1)$	the Gaussian distribution	/	/

Notes: '1': true; '0': false; '/': the value is not a Boolean.

As illustrated in Fig. 4 where the module is supposed to be a powerful module, denoted by m_1, inequalities in Eq. (4) correspond to the six collinear cases (i.e., g_i^c is in X^-, Y^-, Z^-, X^+, Y^+ and Z^+ respectively).

Particularly if $m_i \in B_i(k)$, the directions that are without feedbacks also suggest impossible cases of g_i^c, compared with Eq. (4). After m_i discovers five impossible cases, the remaining case in Eq. (4) should be the correct one. See an example in Fig. 5 where the light colored module represents m_i and the dark colored module represents m_1. On the left-hand side of Fig. 5, $m_i(k) \in B_i(k)$, the red symbol X shows the impossible cases discovered in round k, corresponding to conditions (1), (5), and (6) in Eq. (4). On the right-hand side of Fig. 5, module m_i moves to another cell such that it is again located on the boundary, condition (2) is also checked to be impossible. With further reconfiguration of m_i, it may find the last impossible case and then the remaining case characterizes the property of g_i^c.

3.3. Control strategies

In this section, a distributed strategy will be developed for each ordinary module to converge to a goal configuration. The strategy contains four phases: communicating phase, detecting phase, waiting phase, and updating phase. The pseudocode is presented in Algorithm 1, where the symbols are listed in Table I. In the algorithms, we use k^- to denote the moment state at the beginning of round k (before modules' locomotion). The state at k^- is identical to the state at the end of round $k-1$. Moreover, we use k to denote the moment state at the end of round k (after modules' one-step locomotion).

- *Communicating phase*: Module m_i, $\forall\ i \in S_2$ currently occupying g_i^c sends a message to its adjacent modules (logical value "true").
- *Detecting phase*: Module m_i, $\forall\ i \in S_2$, which neither reaches the goal in Subproblem 2 nor occupies a cell g_i^c, records appropriate feasible cells. That is, if the radial angle $\phi_i(k) = 0$, then all the feasible cells are recorded; otherwise the recorded feasible cells are those belonging to its local polytope as well as not resulting in isolated modules.
- *Waiting phase*: Module m_i, $\forall\, i \in S_2$ with recorded feasible cells in the detecting phase, generates a waiting time according to the following rules. That is, if a module is located on the boundary to its local polytope, it generates a random waiting time within interval (τ_1, τ_2); and if a module is located strictly interior to its polytope, it generates a random waiting time within interval (τ_2, τ_3), where it holds that $0 < \tau_1 < \tau_2 < \tau_3$. Modules then wait for the generated waiting time accordingly.

Algorithm 1 Reconfiguration algorithm for each module $m_i \in S_2$ in round k

```
 1: Input: $\{z^i_j(k^-) : j \in N_i(k^-)\}$
 2: $F_i(k) \leftarrow \emptyset$;
 3: determine $G^c_i$, Achiev$_i(k^-)$, $P_i(k^-)$, $\phi_i(k^-)$ and $f_i(k)$;
 4: Phase I: communicating phase (lines 4-7)
 5: if $z_i(k^-) \in G^c_i$
 6:   send ‘true’ to each adjacent module;
 7: end if
 8: Phase II: detecting phase (lines 8-15)
 9: if Achiev$_i(k^-) = 0$, $z_i(k^-) \notin G^c_i$ and $\phi_i(k^-) = 0$ then
10:   add both $f_i(k)$ to $F_i(k)$;
11: else if Achiev$_i(k^-) = 0$, $z_i(k^-) \notin G^c_i$ and $\phi_i(k^-) \neq 0$ then
12:   for each $f_i(k) \in P_i(k^-)$ s.t. Iso$_i(k) = 0$
13:     add $f_i(k)$ to $F_i(k)$;
14:   end for
15: end if
16: Phase III: waiting phase (lines 16-23)
17: if $F_i(k) \neq \emptyset$
18:   if $z_i(k^-) \in B_i(k^-)$ then
19:     randomly generate $\Delta_i(k) \in (\tau_1, \tau_2)$;
20:   else randomly generate $\Delta_i(k) \in (\tau_2, \tau_3)$;
21:   end if
22:   wait for a period of time $\Delta_i(k)$;
23: end if
24: Phase IV: updating phase (lines 24-33)
25: if $\forall j \in N_i(k^-)$ s.t. LD$(z_i, z_j)|_{k^-} \leq 2$ and $z^i_j(k) = z^i_j(k^-)$ then
26:   randomly generate $\delta_i(k) \in \mathbf{N}(0, 1)$;
27:   if $\delta_i(k) \in (-1.3, 1.3)$
28:     if $z_i(k^-) \in B_i(k^-)$ and exists $f^*_i(k) \in \big(\text{Int}_i(k^-) \cap F_i(k)\big)$ then
29:       $z_i(k) \leftarrow f^*_i(k)$;
30:     else $z_i(k) \leftarrow f^*_i(k), \ \forall f^*_i(k) \in F_i(k)$;
31:     end if
32:   end if
33: end if
```

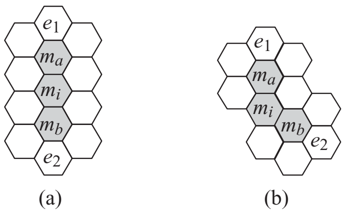


Fig. 6. Nearby distribution of m_i. Grey cells: occupied by modules; e_1 and e_2: either occupied or empty; others: empty.

• *Updating phase*: Module m_i, $\forall\ i \in S_2$, which has waited for the generated waiting time, is a candidate locomoting module. If the module observes that none of its neighbors within 2 lattice distance has moved, it will proceed updating. Depending on whether the module is located on the boundary to its local polytope, it chooses recorded feasible cell that is strictly interior to its local polytope to update.

3.4. Correctness validation

The correctness of Algorithm 1 will be verified in this section. First of all, let us explain the mechanism we adopt to avoid potential isolated modules. Recall the assumption that a module can only detect the nearest module in each direction and we have the following proposition.

Proposition 2. *Suppose $m_i \in S_2$. There would come a potential isolated module from m_i's point of view after locomotion of m_i if and only if m_i's nearby distribution is as illustrated in Fig. 6.*

Proof. First, let us prove the sufficiency. Module m_i in the figure has two adjacent neighbors and it is without motion constraints according to Definition 1. No matter whether e_1 and e_2 are occupied by some modules or not, they are not detectable by m_i as for m_i, module m_a and e_1 (if exists) are located in the same direction, and m_a is nearer than e_1 (similar case for m_b and e_2). From m_i's point of view, both m_a and m_b must regard it as a unique adjacent module. The moving of m_i based on either substrate would potentially make the other become isolated.

Next, let us provide the necessity in a contrapositive way by considering cases of m_i with different number of adjacent modules. (i) Suppose m_i has only one adjacent module. Then the adjacent module should be the unique choice of m_i's substrate and they are still adjacent after locomotion of m_i. (ii) Suppose m_i has at least two adjacent modules, but not like the cases appeared in the figure. The properties of owing feasible cells infer that at most one adjacent module of m_i is uniquely adjacent to m_i again from m_i's point of view. Hence, by letting m_i choose this module as its substrate, no potential isolated module would appear. (iii) Suppose m_i has two adjacent modules m_a and m_b as distributed in the figure. Moreover, at least one of those empty cells (excluding e_1 and e_2) in the figure is occupied by some module. Thus, at least one between m_a and m_b has another adjacent module that is also detectable by m_i, or equivalently, atmost one between m_a and m_b regard m_i as a unique adjacent module from m_i's point of view. By choosing this module as m_i's substrate, no potential isolated module would appear. □

Remark 2. Potential isolated module is measured from a module's local point of view so that sometimes it does not mean a real isolated module from a global point of view. However, avoiding potential isolated modules can well prevent real isolated modules from encountering. Besides, the above proposition presents the case in which a potential isolated module must appear irrespective of the feasible cell choosen. In other words, there exists a choice for a module to choose appropriate feasible cells to avoid potential isolated modules if the nearby distribution of the module is not like the situation illustrated in Fig. 6.

Proposition 3. *Under Algorithm 1, multiple ordinary modules converge to fill up the goal configuration in finite time.*

The key point to prove the finite time convergence is to show that the polytope would decrease with the reconfiguration of modules under the proposed algorithms. Before doing this, we argue that among all possible configurations of N modules with fixed two vertices, the goal configuration has the minimum area (the least number of cells contained in the polytope).

Lemma 1. *Suppose the two powerful modules are collinearly positioned with lattice distance between them equaling $N-1$. The area of polytope reaches its minimum if and only if the goal configuration is achieved.*

Proof. Suppose current configuration is the goal configuration. Then the polytope is a line segment without empty cells. The area of it is N. Polytopes under other configurations contain empty cells, and the area of which must be larger than N.

Now suppose the current configuration is not the goal configuration. Since two powerful modules are fixed, the area of the line segment between them is already equal to N. In addition, several modules are not located on that line segment, the area of polytope is larger than N.

In conclusion, the goal configuration has the minimum polytope area. □

Proof of Proposition 3. For a non-goal configuration, the corresponding polytope is either a polygon or a line segment. First, we will show that the area of a polygon polytope is non-increasing under the proposed algorithms.

Suppose $P(k)$ is a polygon. Therefore, $B(k)$ contains empty cells and we are able to find a module $m_i \in B(k)$ and an empty cell $e_i \in B(k)$ such that m_i and e_i are adjacent. Because m_i is not isolated, there exists another module (denoted by m_j) which is adjacent to m_i.

Without loss of generality, let the line determined by m_i and e_i act as the left boundary of $P(k)$ as shown in Fig. 7 (by rotating the entire configuration $k\pi/3$ radians for some appropriate integer k). Thus, m_j must position on some cell among c_1, c_2, and c_3. Below, by discussing different distributions of the three cells, we will show that there always exists some module capable of moving in a round to reconfigure the modular configuration.

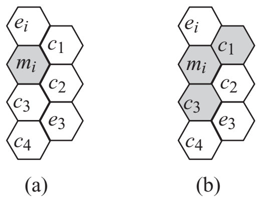


Fig. 7. Feasibility of configuration evolution.

(i) All the three cells $c_1{\sim}c_3$ are occupied by modules (including m_j). Then $e_i{\in}P(k)$ is a feasible cell for m_i, which will be recorded after executing Algorithm 1.

(ii) Two contiguous cells among $c_1{\sim}c_3$ are occupied by modules. Then $e_i{\in}P(k)$ is a feasible cell for m_i when c_1 and c_2 are occupied by modules, and similarly $c_1{\in}P(k)$ is a feasible cell of m_i when c_2 and c_3 are occupied by modules.

(iii) Only one cell among $c_1{\sim}c_3$ is occupied by a module, i.e., m_j. Then $e_1{\in}P(k)$, $c_1{\in}P(k)$, and $c_2{\in}P(k)$ are feasible cells for m_i when c_1, c_2 and c_3 are occupied by m_j respectively.

(iv) Cells c_1 and c_3 are occupied by modules as illustrated in Fig. 7(b). Now suppose e_3 is occupied by some module. Then $e_i{\in}P(k)$ is a feasible cell for m_i. Otherwise, suppose e_3 is empty, which results in that e_i is no longer a feasible cell for m_i due to the mechanism that we adopted to avoid potential isolated modules (the locomotion of m_i from its current cell to e_i would make c_3 a potential isolated module since m_i cannot detect cell c_4). It then follows that $c_2{\in}P(k)$ is a feasible cell for module in c_3 if c_4 is empty, while $e_3{\in}P(k)$ is a feasible cell for module in c_3 if c_4 is not empty.

Based on a polygon polytope, we are always able to find a module capable of moving while guaranteeing the area of current polytope non-increasing as the feasible cells selected are within the current polytope.

Next, suppose $P(k)$ is a line segment. Therefore at least one vertex cell of $P(k)$ is occupied by an ordinary module because current configuration is not a goal configuration. Let m_i represent the ordinary module that acts as the vertex in $P(k)$. Hence, m_i neither has achieved its local objective, nor has positioned on some crude goal cell. Both feasible cells will be recorded after executing Algorithm 1. No matter which feasible cell it moves to, the area of polytope actually increases. However, this would not affect the convergence of the configuration for the following two reasons: (1) The area of a line segment polytope in round k_1 is larger than that of a line segment polytope in round k_2 if $k_1{<}k_2$; (2) once the polytope changes from a line segment to a polygon, the area of which is non-increasing until it reaches to another line segment polytope.

By now the non-increasing of the area of polytope is guaranteed by the proposed algorithms. Non-increasing is not sufficient, and we will further show that the area of polytope would decrease in finite rounds. Algorithm 1 suggests that a module on the boundary to its local polytope has a higher priority to locomote than a module strictly interior. In addition, a boundary module is preferred to move to an interior feasible cell than move to a boundary cell. Hence, the polytope quickly decreases when it is "wide." On the other hand, with the "thinning" of polytope, rich information will be attained for modules to get the idea of equivalent terminating cells. As stated above, the goal configuration is achieved in finite rounds. □

Remark 3. Modules take actions in synchronous rounds, which means that each round lasts the same time period, for example, a constant T. The communicating phase and the detecting phase for each module are assumed to last also for the same time, although in practice different modules may take less time to finish the tasks in each phase. The asynchronous mechanism is caused by the different waiting time generated by different modules. It is deemed that the event that any two different modules generate the same waiting time measures zero in probability. As a result, the concurrent locomotion of modules within one round appears to be in asynchronous manner. Besides, in the updating phase, the Gaussian distribution is introduced which acts as a reference parameter for a module to decide whether to move in a round. The parameter added effectively deals with the situation of "chattering," especially when current configuration approaches the goal configuration. Although it is possible that a configuration is the same as it is in the previous round, the value of the parameter guarantees that a module will move to a new cell with about 80% probability, and will not move with only 20% probability. The probability can also be changed to other values.

4. Convergence of the Entire Configuration

We would like to solve Subproblems 1 and 2 together in this section. The initial positions of all modules are randomly distributed without any isolated module.

It is sure that if Subproblem 1 can be solved within finite rounds, and the finish state of Subproblem 1 is spread to ordinary modules, Subproblem 2 can be reached as discussed in the previous section. Equivalently, the entire modular configuration converges to a straight chain configuration with terminal nodes being powerful modules.

4.1. Control strategies

Different strategies will be proposed for powerful modules and ordinary modules. Both strategies contain four phases similar to the algorithm presented in the previous section. The pseudocodes are presented in Algorithms 2 and 3. Symbols are almost the same as addressed in Table I except that $\text{Achiev}^i_{1,2}$ is a logic value representing module i's understanding of whether the powerful modules have reached goals in Subproblem 1, and $P^i_{1,2}$ is another logic value representing module i's understanding of whether a powerful module is located on the boundary or interior to its local polytope.

Algorithm 2 Reconfiguration algorithm for each module $m_i \in S_1$ in round k

```
1: **Input:** $\{z^i_j(k^-) : j \in N_i(k^-)\}$
2: $F_i(k) \leftarrow \emptyset$;
3: determine $\text{Achiev}_i(k^-)$, $f_i(k)$, $P_i(k^-)$, $\text{LD}(z_i(k^-), \bar{z}_i(k^-))$, $\text{LD}(f_i(k), \bar{z}_i(k^-))$, $\kappa(z_i(k^-), \bar{z}_i(k^-))$, and $\kappa(f_i(k), \bar{z}_i(k^-))$;
4: **Phase I:** communicating phase (lines 4-5)
5: send $\text{Achiev}_i(k^-)$ and $z_i(k^-) \in B_i(k^-)$ (or $\text{Int}_i(k^-)$) to each adjacent module;
6: **Phase II:** detecting phase (lines 6-11)
7: **if** $\text{Achiev}_i(k^-) = 0$ and $\text{LD}(z_i(k^-), \bar{z}_i(k^-)) \neq N-1$ **then**
8:     add each $f_i(k)$ to $F_i(k)$ s.t. $\text{Iso}_i(k) = 0$ and $|\text{LD}(f_i(k), \bar{z}_i(k^-)) - (N-1)| \leq |\text{LD}(z_i(k^-), \bar{z}_i(k^-)) - (N-1)|$;
9: **else if** $\text{Achiev}_i(k^-) = 0$ and $\text{LD}(z_i(k^-), \bar{z}_i(k^-)) = N-1$ **then**
10:     add each $f_i(k)$ to $F_i(k)$ s.t. $\text{LD}(f_i(k), \bar{z}_i(k^-)) = N-1$, $\text{Iso}_i(k) = 0$, and $\kappa(f_i(k), \bar{z}_i(k^-)) \leq \kappa(z_i(k^-), \bar{z}_i(k^-))$;
11: **end if**
12: **Phase III:** waiting phase (lines 12-16)
13: **if** $F_i(k) \neq \emptyset$ **then**
14:     randomly generate $\Delta_i(k) \in (\tau_1, \tau_2)$;
15:     wait for a period of time $\Delta_i(k)$;
16: **end if**
17: **Phase IV:** updating phase (lines 17-26)
18: **if** $\forall j \in N_i(k^-)$ s.t. $\text{LD}(z_i, z_j)|_{k^-} \leq 2$ and $z^i_j(k) = z^i_j(k^-)$ **then**
19:     randomly generate $\delta_i(k) \in \mathbf{N}(0, 1)$;
20:     **if** $\delta_i(k) \in (-1.3, 1.3)$
21:         **if** exists $f^*_i(k) \in F_i(k)$ s.t. $|\text{LD}(f^*_i(k), \bar{z}_i(k^-)) - (N-1)| < |\text{LD}(z_i(k^-), \bar{z}_i(k^-)) - (N-1)|$ **then**
22:             $z_i(k) \leftarrow f^*_i(k)$
23:         **else** $z_i(k) \leftarrow f^*_i(k), \forall f^*_i(k) \in F_i(k)$
24:         **end if**
25:     **end if**
26: **end if**
```

• *Communicating phase*

1. Module m_i, $\forall\, i \in S_1$, sends a message to its adjacent modules containing the information of whether it has reached the goal in Subproblem 1, and whether it is located on the boundary or interior to its local polytope.

2. Module m_i, $\forall\, i \in S_2$, sends a message to its adjacent modules, including module i's understanding about the status of powerful modules, i.e., to its knowledge, whether a powerful module has reached the goal in Subproblem 1, and whether a powerful module is located on the boundary or interior to its local polytope. Moreover, module i observes that the information of whether it satisfied the equivalent terminating condition is also included in the message.

• *Detecting phase*

1. Module m_i, $\forall\, i \in S_1$, which has not reached the goal in Subproblem 1, records appropriate feasible cells. That is, if the lattice distance between the two powerful modules is equal to $N - 1$, then record

Algorithm 3 Reconfiguration algorithm for each module $m_i \in S_2$ in round k

```
1: Input: $\{z_j^i(k^-) : j \in N_i(k^-)\}$
2: $F_i(k) \leftarrow \emptyset$;
3: determine $G_i^c$, $\mathrm{Achiev}_i(k^-)$, $P_i(k^-)$, $\phi_i(k^-)$, $f_i(k)$, $\mathrm{Achiev}_{1,2}^i(k^-)$ and $P_{1,2}^i(k^-)$;
4: Phase I: communicating phase (lines 4-11)
5: if $\mathrm{Achiev}_{1,2}^i(k^-) = 1$
6:   if $z_i(k^-) \in G_i^c$ then
7:     send $\mathrm{Achiev}_{1,2}^i(k^-) = 1$ and $z_i(k^-) \in G_i^c$ to each adjacent module;
8:   else send $\mathrm{Achiev}_{1,2}^i(k^-) = 1$ to each adjacent module;
9:   end if
10: else send $\mathrm{Achiev}_{1,2}^i(k^-) = 0$ and $P_{1,2}^i(k^-)$ to each adjacent module;
11: end if
12: Phase II: detecting phase (lines 12-19)
13: if $\mathrm{Achiev}_{1,2}^i(k^-) = 0$, $z_i(k^-) \in B_i(k^-)$ and $z_{1,2} \in \mathrm{Int}_{1,2}(k^-)$ then
14:    for each $f_i(k) \notin \mathrm{Int}_i(k^-)$ s.t. $\mathrm{Iso}_i(k) = 0$
15:      add $f_i(k)$ to $F_i(k)$;
16:    end for
17: else
18:   run lines 9-15 in Algorithm 1;
19: end if
20: Phase III: waiting phase (lines 20-27)
21: if $F_i(k) \neq \emptyset$
22:   if $\mathrm{Achiev}_{1,2}^i(k^-) = 1$ and $z_i(k^-) \in B_i(k^-)$ then
23:     randomly generate $\Delta_i(k) \in (\tau_1, \tau_2)$;
24:   else randomly generate $\Delta_i(k) \in (\tau_2, \tau_3)$;
25:   end if
26:   wait for a period of time $\Delta_i(k)$;
27: end if
28: Phase IV: updating phase (lines 28-41)
29: if $\forall j \in N_i(k^-)$ s.t. $\mathrm{LD}(z_i, z_j)|_{k^-} \leq 2$ and $z_j^i(k) = z_j^i(k^-)$ then
30:   randomly generate $\delta_i(k) \in \mathbf{N}(0,1)$;
31:   if $\delta_i(k) \in (-1.3, 1.3)$
32:     if $\mathrm{Achiev}_{1,2}^i(k^-) = 1$
33:       run lines 28-31 in Algorithm 1;
34:     else
35:       if $z_i(k^-) \in B_i(k^-)$, $z_{1,2} \in \mathrm{Int}_{1,2}(k^-)$ and exists $\{f_i^*(k) : (f_i^*(k) \in F_i(k)) \cap (f_i^*(k) \notin P_i(k^-))\}$ then
36:         $z_i(k) \leftarrow f_i^*(k)$;
37:       else $z_i(k) \leftarrow f_i^*(k), \forall f_i^*(k) \in F_i(k)$;
38:       end if
39:     end if
40:   end if
41: end if
```

those feasible cells that do not change the lattice distance between the powerful modules, reduce the degree of collinearity between them, and do not bring isolated modules. If the lattice distance between the powerful modules does not equal to $N-1$, record those feasible cells that make the lattice distance between the powerful modules change monotonically toward $N-1$, and do not bring isolated modules.

2. For a module m_i, $i \in S_2$, if it lies on the boundary of its own local polytope and knows that the powerful modules have not reached the goal in Subproblem 1 and a powerful module is located in the interior of powerful module's local polytope, then record those feasible cells that are either located on the boundary or outside module m_i's local polytope. Otherwise, do the same things as in Algorithm 1.

• *Waiting phase*

1. Module m_i, $\forall\, i \in S_1$, with recorded feasible cells randomly generates a waiting time within interval (τ_1, τ_2) and then waits for the generated waiting time accordingly.

2. Module m_i, $\forall\, i \in S_2$, with recorded feasible cells in the detecting phase, generates a randomly waiting time according to the status of powerful modules to its knowledge. That is, if the powerful modules have not reached the goal in Subproblem 1 to its knowledge, it generates a random waiting

time within interval (τ_2, τ_3). Otherwise, do the same things as in Algorithm 1. Modules then wait for the generated waiting time accordingly.

• *Updating phase*: Module m_i, which has waited for the generated waiting time, is a candidate locomoting module. If the module observes that none of its neighbors within 2 lattice distance has moved, it will proceed updating.
1. If $m_i \in S_1$, a recorded feasible cell which makes the lattice distance between the powerful modules change toward $N - 1$ is preferred.
2. If $m_i \in S_2$ and to its knowledge, the powerful modules have not reached the goal in Subproblem 1, a recorded feasible cell which is outside $m_i's$ local polytope is preferred. Otherwise an ordinary module does the same thing as in Algorithm 1.

4.2. Correctness validation

The correctness of Algorithms 2 and 3 will be discussed in this section by respectively showing the finite time convergence of two powerful modules and finite time convergence of the entire configuration.

Proposition 4. *Under Algorithm 2 and Algorithm 3, two powerful modules converge to achieve the desired distribution in finite time.*

Proof. Powerful modules have the highest locomotion priority as addressed in the proposed algorithms. Consider a powerful module m_1. If m_1 records some feasible cell, the moving of m_1 would make the lattice distance between two powerful modules approaching $N-1$ (at least not increasing). Otherwise if m_1 cannot locomote in a round although it has not achieved local objective, it will be able to move some rounds later after reconfiguration of some ordinary modules. This is because the information about the achievement state of powerful modules will be received by adjacent modules, the behavior of which will relax the motion constraints encountered by powerful modules. For the modules without receiving such information, they are not adjacent to the ones received, and hence will not affect the motion of the modules that received the information. With the help of some ordinary modules, the two powerful modules would converge to achieve desired distribution, i.e., collinearly positioned with lattice distance $N-1$ in finite time. □

Proposition 5. *Under Algorithm 2 and Algorithm 3, the modules converge to a goal configuration in finite time.*

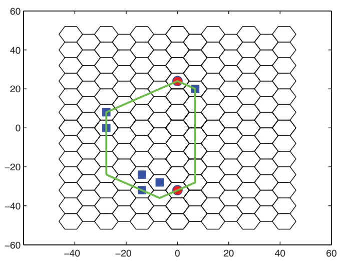
Proposition 5 is obtained by combining the analysis for Propositions 4 and 3. Hence, we do not restate details here.

Remark 4. From a global point of view, the goal configuration is fixed under Algorithm 1 since the positions of two powerful modules are initially fixed. However, ordinary modules are freely occupying any middle cells in a goal configuration. The goal configuration under Algorithm 2 and Algorithm 3 is not fixed due to the uncertainty of final positions of powerful modules, although the relative position between them is known (desired distribution).

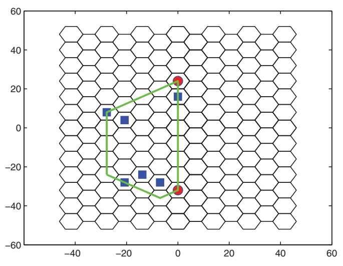
5. Simulation and Discussion

Simulations are done using MATLAB to validate the proposed algorithms. Fig. 8 shows an example of reconfiguration of ordinary modules while the two powerful modules are initially set at desired positions (validating conclusions obtained in Section 3). Ordinary modules finally achieve their local objectives after 27 rounds. From Fig. 8(g), we observe that the area of the enclosing polytope is $41 \to 34 \to 26 \to 21 \to 15 \to 8$ as time goes on, which is non-increasing with respect to the evolution of modules and strictly decreases after a finite number of rounds.

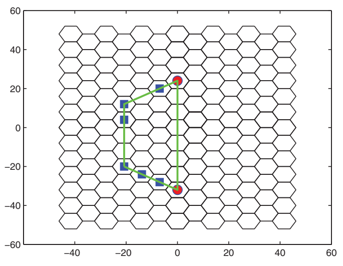
Two tests under Algorithms 2 and 3 are illustrated in Figs. 9 and 10 where the initial lattice distance between two powerful modules is less and larger than the desired value respectively. Besides initial and goal configurations, six intermediate configurations are also presented, each shows the change of area of polytopes during the reconfiguration process. According to Fig. 9(g), we discover that (i) the lattice distance between powerful modules is increasing until it reaches 7 (round 11); (ii) the collinearity degree of powerful modules is decreasing since round 11 until it reaches zero (round 14); and (iii) after powerful modules have achieved local objective (round 14), the area of polytope



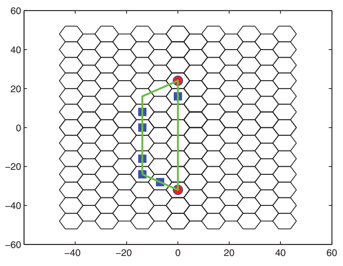
(a)



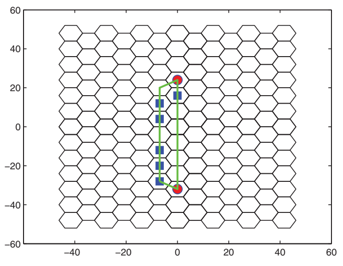
(b)



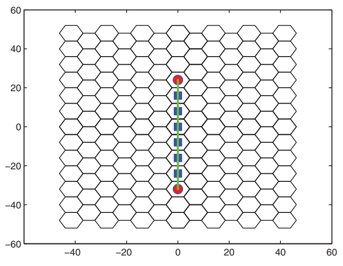
(c)



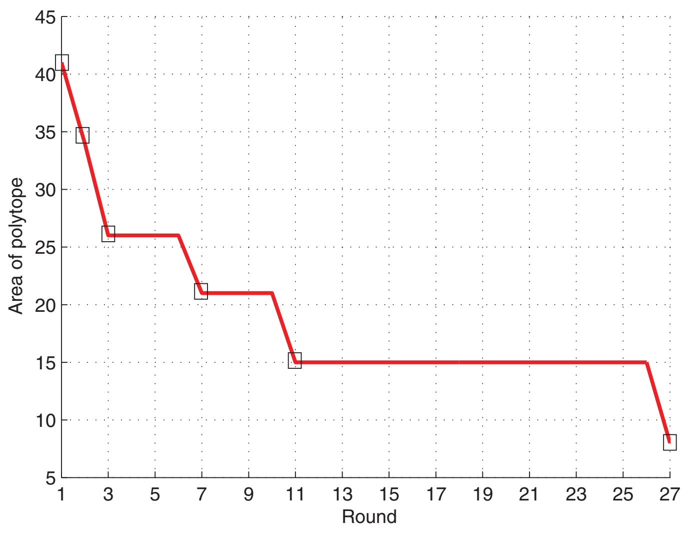
(d)



(e)



(f)



(g)

Fig. 8. (Colour online) Example illustrating convergence of ordinary modules. (a) Initial configuration; (b)–(e) several intermediate rounds; (f) goal configuration; (g) response of area of polytope. (a)–(f) are matching with the rounds marked with black rectangular in (g). Red circle: powerful modules; blue rectangle: ordinary modules; green line: boundary of polytope.

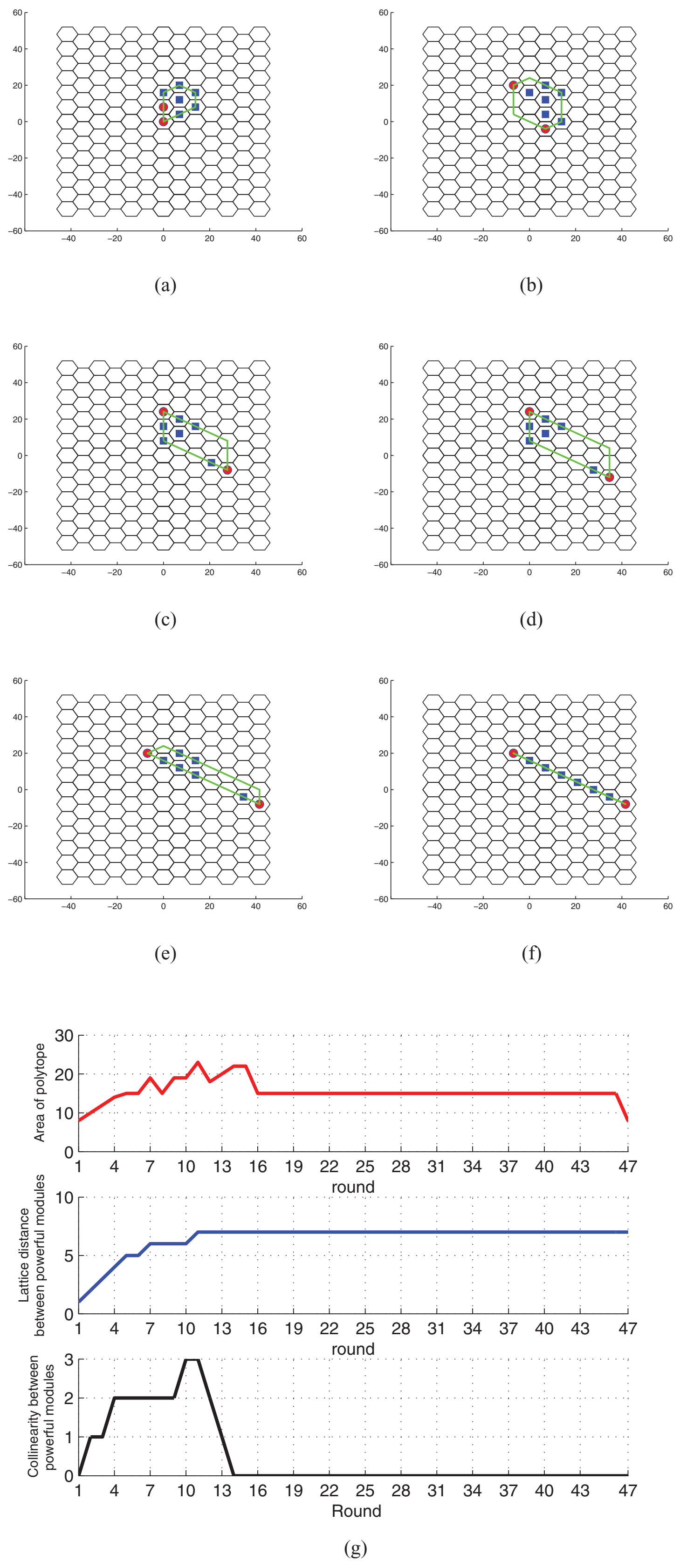
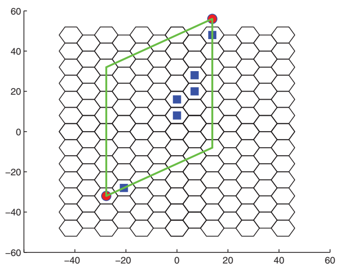
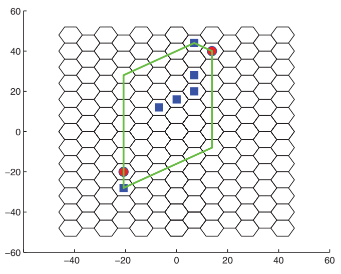


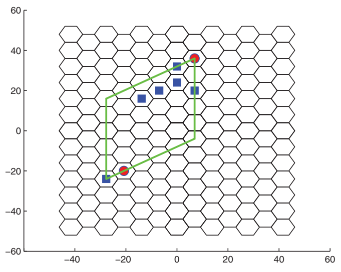
Fig. 9. (Colour online) Example illustrating convergence of all modules. (a) Initial configuration; (b)–(e): several intermediate rounds; (f) goal configuration; (g) (from top to bottom) area of polytope; lattice distance between powerful modules; collinearity between powerful modules. Red circle: powerful modules; blue rectangle: ordinary modules; green line: boundary of polytope.



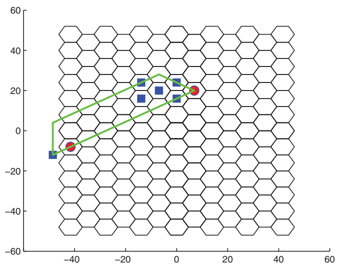
(a)



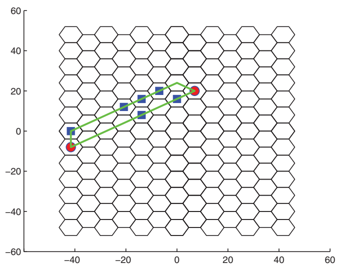
(b)



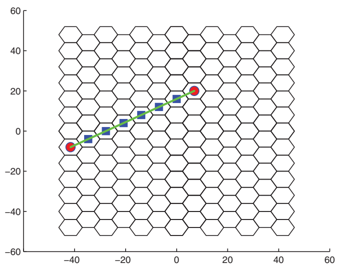
(c)



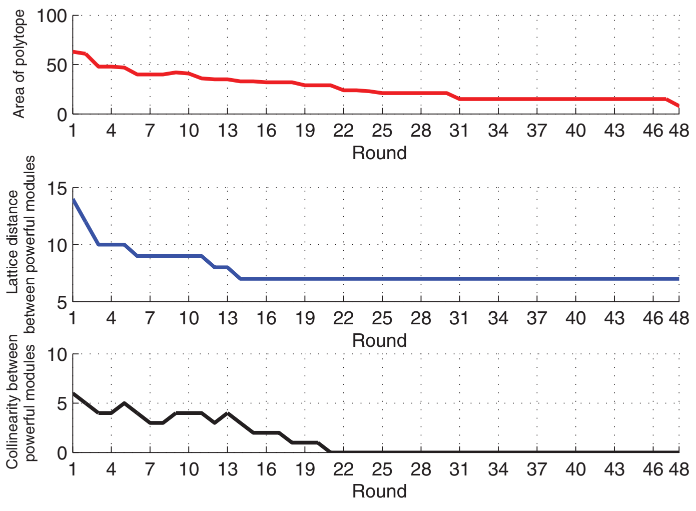
(d)



(e)



(f)



(g)

Fig. 10. (Colour online) Another example illustrating convergence of all modules. (a) Initial configuration; (b)–(e) several intermediate rounds; (f) goal configuration; (g) (from top to bottom) area of polytope; lattice distance between powerful modules; collinearity between powerful modules. Red circle: powerful modules; blue rectangle: ordinary modules; green line: boundary of polytope.

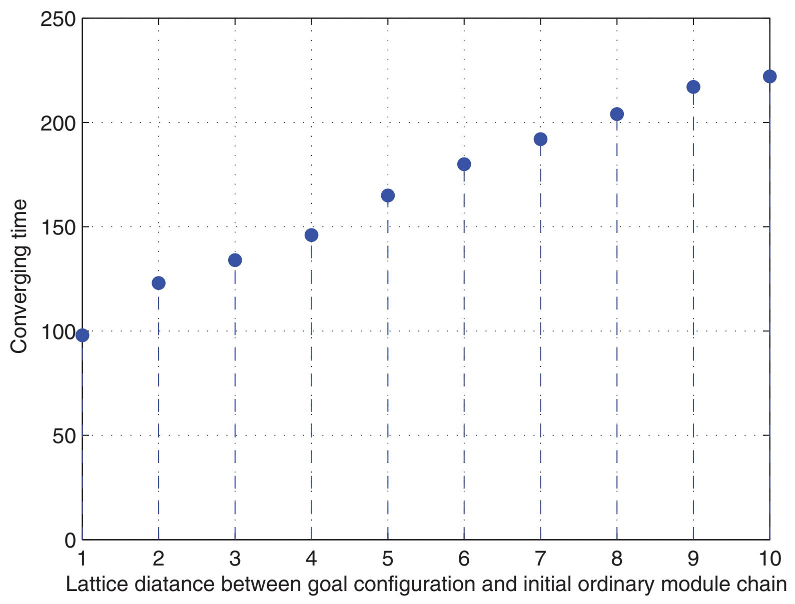


Fig. 11. (Colour online) Relationship between the converging time and the initial lattice distance between the desired goal configuration and the initial ordinary module chain.

is decreasing until the goal configuration is achieved (round 47). Similarly, according to Fig. 10(g), we find that (i) the lattice distance between powerful modules is decreasing until reaches 7 (round 14); (ii) the collinearity degree of powerful modules is decreasing since round 14 until reaches zero (round 21); and (iii) after the powerful modules have achieved their local objective (round 21), the area of the enclosing polytope is decreasing until the goal configuration is achieved (round 48). Both results coincide well with the conclusions we obtained in Section 4.

Moreover, we discover that under Algorithm 1(i) the converging time for the case when ordinary modules are initially dispersed is less than that for the case when ordinary modules are initially densely positioned, and (ii) it takes less time for an initial configuration to approach a near-goal configuration (in which each ordinary module at most departs from crude goal cells with the lattice distance 1), but the convergence from a near-goal configuration to the goal configuration usually spends much more time. An example is shown in Fig. 11, in which the ordinary modules are initially collinear and the line segment is parallel to the desired goal configuration. The figure shows the relationship between the converging time (number of rounds) and the initial lattice distance between the desired goal configuration and the initial ordinary module chain (the converging time under each case is an average value that is obtained by doing about 50 tests).

6. Conclusions and Future Work

We have developed a distributed strategy for a group of modules with limited abilities to transform from arbitrary initial configuration to a straight chain. Based on the idea of curve shortening, the strategy well guarantees the convergence to a goal configuration within finite rounds.

The key ideas of this study for solving the reconfiguration problem toward a straight chain fall into two aspects: Use of as less as possible powerful modules and use of as more as possible ordinary modules require only local measurements and local communications, which can be made with low costs. Based on the geometric property of the goal configuration (straight chain), decompose the reconfiguration problem into two subproblems which can be realized locally. The algorithms presented in this study are specifically for a goal configuration of straight chain. However, the ideas can be extended to goal configurations of other shapes. For example, for an "8"-shaped goal configuration, we can use three powerful modules that are expected to occupy the top, middle, and bottom cells such that the three cells are collinear and spaced equally. Since an "8" can be regarded as two connected circles (regular hexagons in a hexagonal lattice), the geometric property of the two circles can be described as to have even curvatures at any point on the circle, or equivalently, to have a uniform lattice distance to the center of the circle. For an S-shaped goal configuration, we can use four powerful modules. Similar geometric properties can be extracted, and distributed reconfiguration schemes can be developed accordingly. For a general goal configuration, the concrete procedures would be much complicated. This is left for the future work, including further investigating general principles to solve reconfiguration problems in a distributed manner.

Acknowledgment

The work is supported by National Natural Science Foundation of China under Grant 61273113.

References

1. C. J. Chiang and G. S. Chirikjian, "Modular robot motion planning using similarity metrics," *Auton. Robots* **10**(1), 91–106 (2001).
2. G. Chirikjian, A. Pamecha and I. Ebert-Uphoff, "Evaluating efficiency of self-reconfiguration in a class of modular robots," *J. Robot. Syst.* **13**(5), 317–338 (1996).
3. M. Yim, Y. Zhang, J. Lamping and E. Mao, "Distributed control for 3D metamorphosis," *Auton. Robots* **10**(1), 41–56 (2001).
4. K. Stoy, "Controlling Self-Reconfiguration Using Cellular Automata and Gradients," **In**: *Proceedings of the 8th International Conference on Intelligent Autonomous Systems* (2004) pp. 693–702.
5. K. Harada, D. Oetomo, E. Susilo, A. Menciassi, D. Daney, J. P. Merlet and P. Dario, "A reconfigurable modular robotic endoluminal surgical system: Vision and preliminary results," *Robotica* **28**, 171–183 (2010).
6. G. S. Chirikjian, "Kinematics of a Metamorphic Robotic System," **In**: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (1994) pp. 449–455.
7. S. Murata, H. Kurokawa and S. Kokaji, "Self-Assembling Machine," **In**: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (1994) pp. 441–448.
8. A. Pamechal, I. E. Uphoff and G. S. Chirikjian, "Useful metrics for modular robot motion planning," *IEEE Trans. Robot. Autom.* **13**(4), 531–545 (1997).
9. J. E. Walter, J. L. Welch and N. M. Amato, "Concurrent metamorphosis of hexagonal robot chains into simple connected configurations," *IEEE Trans. Robot. Autom.* **18**(6), 945–956 (2002).
10. J. E. Walter, J. L. Welch and N. M. Amato, "Distributed reconfiguration of metamorphic robot chains," *Distrib. Comput.* **17**(2), 171–189 (2004).
11. R. Ghrist and V. Peterson, "The geometry and topology of reconfiguration," *Adv. Appl. Math.* **38**(3), 302–323 (2007).
12. L. Zhang and J. S. Dai, "Metamorphic Techniques and Geometric Reconfiguration Principles," **In**: *International Conference on Reconfigurable Mechanisms and Robots 2009* (2009) pp. 32–40.
13. S. Matysik and J. Walter, "Using a Pocket-Filling Strategy for Distributed Reconfiguration of a System of Hexagonal Metamorphic Robots in an Obstacle-Cluttered Environment," **In**: *Proceedings of the 2009 IEEE International Conference on Robotics and Automation* (2009) pp. 3266–3273.
14. J. E. Walter, E. M. Tsai and N. M. Amato, "Algorithms for fast concurrent reconfiguration of hexagonal metamorphic robots," *IEEE Trans. Robot.* **21**, 621–631 (2005)
15. F. Hou and W. Shen, "On the Complexity of Optimal Reconfiguration Planning for Modular Reconfigurable Robots," **In**: *2010 IEEE International Conference on Robotics and Automation* (2010) pp. 2791–2796.
16. E. Guan, Z. Fu, W. Yan, D. Jiang and Y. Zhao, "Self-reconfiguration path planning design for M-lattice robot based on genetic algorithm," *Intell. Robot. Appl.* **7102**, 505–514 (2011).
17. P. Ivanov and J. Walter, "Layering Algorithm for Collision-Free Traversal Using Hexagonal Self-Reconfigurable Metamorphic Robots," **In**: *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010) pp. 521–528.
18. Y. Miao, G. Yan and Z. Lin, "A Distributed Reconfiguration Strategy for Target Enveloping with Hexagonal Metamorphic Modules," **In**: *Proceedings of the 2011 IEEE International Conference on Robotics and Automation* (2011) pp. 4804–4809.
19. T. Larkworthy and S. Ramamoorthy, "A characterization of the reconfiguration space of self-reconfiguring robotic systems," *Robotica* **29**(1), 73–85 (2011).
20. A. Pamecha, C. chiang, D. Stein and G. Chirikjian, "Design and Implementation of Metamorphic Robots," **In**: *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference* (1996) pp. 1–10.
21. S. Murata, H. Kurokawa, E. Yoshida, K. Tomita and S. Kokaji, "A 3-D Self-Reconfigurable Structure," **In**: *Proceedings of the 1998 IEEE International Conference on Robotics and Automation* (1998) pp. 432–439.
22. A. Castano, W. M. Shen and P. Will, "CONRO: Towards deployable robots with inter-robots metamorphic capabilities," *Auton. Robots* **8**(3), 309–324 (2000).
23. B. Salemi, M. Moll and W. Shen, "SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System," **In**: *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2006) pp. 3636–3641.

24. K. Gilpin, K. Kotay, D. Rus and I. Vasilescu, “Miche: Modular shape formation by self-disassembly,” *Int. J. Robot. Res*. **27**(3–4), 345–372 (2008).
25. K. Gilpin, A. Knaian and D. Rus, “Robot Pebbles: One Centimeter Modules for Programmable Matter Through Self-Disassembly,” **In**: *Proceedings of the 2010 IEEE International Conference on Robotics and Automation* (2010) pp. 2485–2492.
26. M. A. Grayson, “The heat equation shrinks embedded plane curves to round points,” *J. Differ. Geom*. **26**(2), 285–314 (1987).
27. M. A. Grayson, “Shortening embedded curves,” *Ann. Math*. **129**, 71–111 (1989).
28. D. Yamins and R. Nagpal, “Automated Global-to-Local Programming in 1-D Spatial Multi-Agent Systems,” **In**: *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (2008) 615–622.