An Approach to the Bio-Inspired Control of Self-reconfigurable Robots

Dongyang Bie[1], Miguel A. Gutiérrez-Naranjo[2], Jie Zhao[1], and Yanhe Zhu[1(✉)]

[1] State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin, Heilongjiang, China
yhzhu@hit.edu.cn

[2] Department of Computer Science and Artificial Intelligence, University of Seville, Seville, Spain
magutier@us.es

Abstract. Self-reconfigurable robots are robots built by modules which can move in relationship to each other. This ability of changing its physical form provides the robots a high level of adaptability and robustness. Given an initial configuration and a goal configuration of the robot, the problem of self-regulation consists on finding a sequence of module moves that will reconfigure the robot from the initial configuration to the goal configuration. In this paper, we use a bio-inspired method for studying this problem which combines a cluster-flow locomotion based on cellular automata together with a decentralized local representation of the spatial geometry based on membrane computing ideas. A promising 3D software simulation and a 2D hardware experiment are also presented.

1 Introduction

Modular Self-reconfigurable (MSR) robots [12] are robots built by modules which can move in relationship to each other. This ability can change its physical form and provide MSR robots a high level of adaptability and robustness [32]. The modularity allows the robot to optimize their shape for different tasks and the control of the movement of the modules represents a big challenge for the development of new research ideas [26]. In fact, such control of the modules and the locomotion planning is a complex non-linear problem and there is no analytic solution for it. According to the existence of center controller or not, current approaches can be generally divided into two categories: centralized control and decentralized control. The problem of obtaining a centralized control is a NP problem [14], so decentralized approaches are currently on the focus of many research approaches in order to achieve effective solutions.

The decentralized control of self-reconfiguration has been studied from different points of view. One of the most interesting is to consider nature as a source of inspiration. In the literature, several bio-inspired methods have been applied for the distributed control of self-reconfigurable robots, among them, methods based on cellular automata (CA) [9,35] or particle swarm optimization [34] can

be cited. They all have the distributed nature for emergent systems from bottom interaction to global regular phenomenon. This match of decentralized character can contribute to the scalability of module numbers by focusing on local agents, but the convergence problem still stands out in the emergent process of swarm systems.

Recently, a bio-inspired approach based on ideas taken from membrane computing and CA has been presented [5]. In this paper, we go on with the idea of combining a cluster-flow locomotion based on CA together with a decentralized local representation of the spatial geometry based on membrane computing ideas. The used method represents a novelty in the framework of self-configurable robots in a double sense, from a theoretical and practical point of view. From a theoretical side, an abstract representation of the robot beyond its physical representation is considered. From a practical point of view, the solution is based on two of the basic features of one of the most studied membrane computing devices, the so-called cell-like P systems: On the one hand, the tree-like graph structure which can be abstracted from the hierarchical arrangement of vesicles in an alive eucaryotic cell. On the second hand, in membrane computing the information is encapsulated in vesicles and encoded by multisets of simple objects. The key point for the use of such multisets in the framework of self-configurable robots is the *interpretation* of the objects. As it will be pointed out below, such objects can represent the length, the relative angle of a module of the robot or any other feature chosen by the designer.

Next, we briefly recall some ideas about the bio-inspired computational research areas used in this paper, namely, membrane computing and CA. Membrane computing [20] is inspired by the structure and functioning of cells as living organisms able to process and generate information. In particular, it focuses on membranes, which are involved in many reactions taking place inside the cell. The basic idea is inspired in the flow of metabolites between cells of a living tissue or between the organelles in an eucaryotic cell. This flow of metabolites can be interpreted as a flow of information for computational purposes. Membrane computing devices are called P systems. They are distributed and have a high degree of parallelism. Such degree of autonomy and the possibility of locally encapsulate the local information needed for the next step of computation make these devices suitable for modelling the geometry of modular self-reconfigurable robots. The second bio-inspired tool used in this paper, CA [30], has been widely used in the literature for the control of self-reconfigurable robots. CA were introduced to decentralized control of MSR robots by Butler *et al.* [9]. Since then, many other approaches can be found in the literature (e.g., [7,10]). In this paper, CA are used to handle the distributed and parallel motion of decentralized modules in MSR robots.

The paper is organized as follows: Firstly, we recall some basics on MSR robots and membrane computing. We present how the cell-like structure of a P system can be interpreted as a configuration of a MSR. Next, we show how the configuration of a tree-like structure can be geometrically represented by a self-reconfigurable robot. Such representation is performed by a cluster-flow

locomotion of spare modules inspired on the well-known turtle graphics methods. This is illustrated with a 3D software simulation and a 2D hardware experiment. Finally, the paper finishes with some conclusions and open research lines.

2 MSR Robots

There are several categories of hardware architecture for MSR robotic modules, such as lattice architecture (see, e.g., [22,28]), mobile architecture (e.g., [17,27,33]), chain architecture (e.g., [2,29]) and hybrid architecture (e.g., [6,31,35]). For the regular geometric organization, lattice architecture is the most convenient for computer modeling among all those architectures. As illustrated in Fig. 1, each module in a lattice architecture has a cubic structure with local coordinate. This module is also called sliding cube model (SCM) module, which has been used as a common module for rapid verification of control methods. Each module is a completely independent working robot with three kinds of basic motion ability: convex motion, linear motion and concave motion. A MSR robot with multi SCM modules changes its global topology by adjusting relative relationships of inner modules through a self-organizing process.

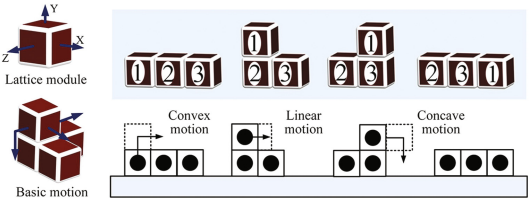

Fig. 1. A 3D structure made with blocks.

The configuration of the whole robot is determined by the relative position of inner modules. Since the independent modules have various possible motion plans, choosing an appropriate mechanism for controlling the movement of each module is a hard task [15,32]. Bearing in mind that each lattice module has six connecting faces and directly connected modules has four relative connecting

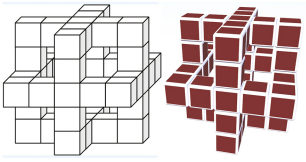

Fig. 2. A schematic representation of a 3D structure made with 42 blocks and its associated robotic structure made with modules.

orientations, a robot system with n modules has $n^{6\times4}$ kinds of connecting ways. Such amount of possibilities makes difficult the control of the robot even for robots with a low amount of modules, as the structure shown in Fig. 2, where a robotic structure with 42 modules is presented.

3 Membrane Computing

According to their topology[1], there are three basic sets of P system models, although other approaches are possible [20]: cell-like P systems, where membranes have a tree-like structure; tissue-like P systems, where membranes are placed in the nodes of a general graph; and spiking neural P systems, which are inspired by the structure of living neurons in a brain. In this paper we use cell-like P systems in order to represent the spatial geometry of MSR.

The basic cell-like P system model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into regions, that contain multisets of objects (represented by symbols of an alphabet) and/or other membranes. The intuition behind this membrane structure is taken from biology. A membrane can be seen as a three-dimensional vesicle which is a separator of the region *inside* and the region *outside*. Biological metabolites inside the regions are modelled by object-symbols. Each region, which is defined by a membrane, can contain other symbols or other membranes, so that a P system has exactly one outer membrane, called the skin membrane, and a hierarchical relationship governing all its membranes under the skin membrane. The *information* encapsulated inside each region is encoded in the type of symbols, but also in its multiplicity. In this paper, the structure of cell-like P systems is used to construct branching structures of self-reconfigurable robots. The rooted tree nature of membranes is a perfect frame to encode the branching structure. In this paper, we use the formal framework of membrane computing in order to describe the geometry and the topology of MSR robots whose modules can be represented with a tree-like structure. The key points of the representation are the following:

1. Firstly, the geometrical structure of each segment (concerning to length, thickness, color or whatever other features) is represented by a multiset of objects placed in the corresponding membrane.
2. Secondly, the topological relations among the segments are represented by the tree-like membrane structure of the P system. If two segments are joint in the robot, the corresponding membranes are joint in the tree-like structure of the P system, i.e., one of them is contained in the other one.
3. Thirdly, the relative position of a module with respect to its father in the segment will be also encoded with a multiset of objects placed inside the corresponding membrane.

[1] Since there are an extensive literature on the use of CA for the control of self-reconfigurable robots, we focus on the membrane computing ideas used in this paper.

The encapsulation of the information in P systems makes possible a natural translation of the idea of module from a physical robot to the formal computational model. One of the main advantages of this formalism is that no global position is needed in order to describe the topology or geometry of the robot.

3.1 An Example

As an initial example, let us consider the figure composed by six Greek crosses as shown in Fig. 2. Each of the crosses is composed by two bars of five cubic modules. The modules at the end of the bars are shared by two crosses and the whole figure has cubical symmetry. Such figure can be thought as the composition of 42 cubical modules which can be distributed into 24 solid segments of length units 1, 2 or 3. By keeping the topological connection of (at least) one of ends of each bar, the 24 solid segments of the figure can be *unrolled* in a like-tree planar figure shown in Fig. 3(left). Let us remark that the relative position between adjacent segments in the planar representation is the same than in the original 3D figure in the following sense: *If two segments are adjacent in the planar representation, they are also adjacent in the 3D figure.*

In Fig. 3(left), a label from in $a, b, \ldots, x$ is associated to each segment and a dual representation of this tree-like structure is depicted in Fig. 3(right). In this new representation, segments of the planar structure are represented by nodes in the graph and there is an edge between the nodes x and y if and only if the segments x and y are joint in the planar representation of the figure. This dual representation will be used for representing the topology of the robot as a cell-like P system membrane structure. The correspondence is immediate since such membrane structure is also a tree-like arrangement of membranes. Figure 4(left)

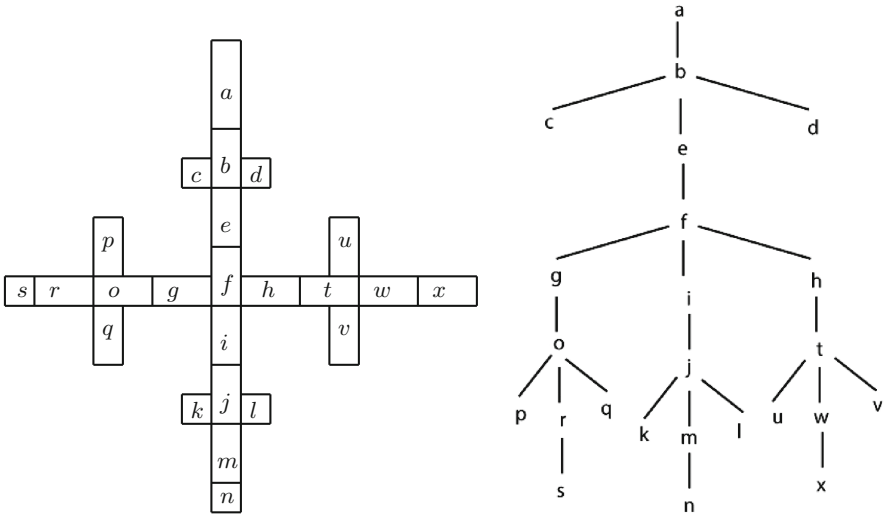

Fig. 3. Tree structure for inner relationships of the structure in Fig. 2.

shows a cell-like P system structure associated to the structure in Fig. 3. Let us remark that this cell-like P system structure takes the vertex a as the root of the tree and hence, the corresponding membrane in the P system structure is the *skin* of the P system, but any other terminal node could be taken as root.

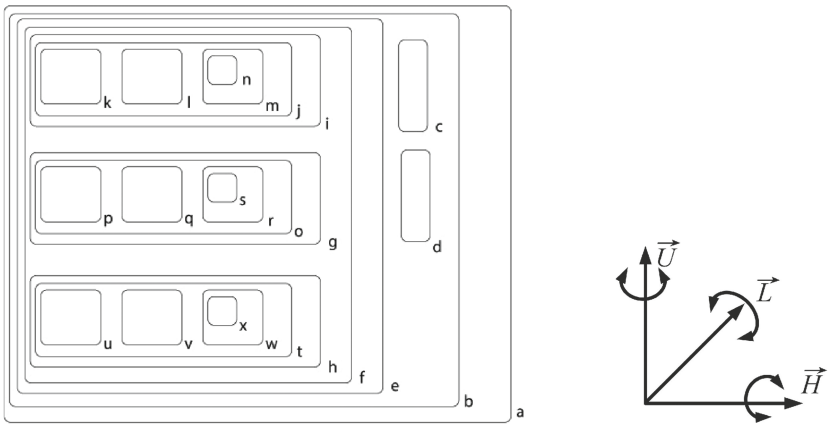

Fig. 4. Membrane structure of a P systems representing the topological structure of the tree in Fig. 3(Left). Rotation around axis (Right).

The tree-like graph in Fig. 3(right) can be immediately obtained from the P system membrane structure in Fig. 4(left), but if we want to represent the original structure from Fig. 2, several symbols must be placed in the membrane structure which encode the geometric features. Such combination of membrane structure plus the symbols associated to each membrane is called a *configuration* in membrane computing. In this example, we choose the symbol F for representing a length unit. For the sake of simplicity, in this example the unique feature of the segment of the robot described by symbols is the length. Nonetheless, other features (width, color, ...) can also been described by multisets of symbols. According to the membrane computing theory, several copies of a symbol can appear in a membrane. The number of copies of F in a membrane will represent the length of the segment associated to the number of copies of F. Instead of a global position of each segment, a representation of the *relative position* of a segment with respect to its father in the tree-like representation is proposed. In this way, some ideas are borrowed from [1,19].

Let us consider the set of vectors $(\boldsymbol{H}, \boldsymbol{L}, \boldsymbol{U})$, with unit length, perpendicular to each other and satisfying $\boldsymbol{H} \times \boldsymbol{L} = \boldsymbol{U}$. A new vector $(\boldsymbol{H}', \boldsymbol{L}', \boldsymbol{U}')$ can be obtained from the vector $(\boldsymbol{H}, \boldsymbol{L}, \boldsymbol{U})$ by using a rotation matrix $\mathbf{R}$ with the

composition $(\boldsymbol{H}', \boldsymbol{L}', \boldsymbol{U}') = (\boldsymbol{H}, \boldsymbol{L}, \boldsymbol{U})\mathbf{R}$. Rotations by angle α about vectors $\boldsymbol{U}$, $\boldsymbol{L}$ and $\boldsymbol{H}$ are represented by Eq. 1 ($c\alpha = \cos\alpha$, $s\alpha = \sin\alpha$).

$$\mathbf{R}_U(\alpha) = \begin{pmatrix} c\alpha & s\alpha & 0 \\ -s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}_L(\alpha) = \begin{pmatrix} c\alpha & 0 & -s\alpha \\ 0 & 1 & 0 \\ s\alpha & 0 & c\alpha \end{pmatrix} \quad \mathbf{R}_H(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{pmatrix} \tag{1}$$

Figure 4(right) illustrates the rotation around the axis. For the regular organization of lattice modules, the angle α is set to be $\alpha = \pi/2$. But the construction can be made in general. By fixing $\alpha = \pi/2$, we have these rotating matrices:

$$\mathbf{R}_U = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}_L = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \mathbf{R}_H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{2}$$

Since all rotating angles between lattice modules are $\pi/2$ rotations, it suffices to introduce the symbols $\mathbf{R}_L$, $\mathbf{R}_U$ and $\mathbf{R}_H$ into the membranes. The occurrence of such symbol in a membrane will be interpreted as the rotation angle of the corresponding robot segment with respect to its segment father in the like-tree structure. In a similar way that with the F symbol, the multiplicity has also a associated meaning. In a natural way, we will consider that the rotation is applied as many times as the number of copies. In such way, we can represent rotations of $\pi/2$, π or $-\pi/2$ by considering one, two or three copies of the symbol.

In this way, the general position of the 3D structure of Fig. 2 can be encoded in a P system configuration by considering the membrane structure shown in Fig. 4(left) and adding to the membrane i the following multiset of symbols w_i (symbol w_i describes the multiset of objects in the membrane associated to the segment i in Fig. 3, and relative orientations are organized respect to the parent membrane, the superscripts [20] denote the multiplicity):

$$\begin{array}{llll} w_a = F^3 & w_g = F^2\,\mathbf{R}_H & w_m = F^2 & w_s = F\,\mathbf{R}_U \\ w_b = F^2\,\mathbf{R}_U & w_h = F^2\,\mathbf{R}_H^3 & w_n = F\,\mathbf{R}_U & w_t = F^2\,\mathbf{R}_U \\ w_c = F\,\mathbf{R}_H & w_i = F^2 & w_o = F^2\,\mathbf{R}_U & w_u = F^2\,\mathbf{R}_H^3 \\ w_d = F\,\mathbf{R}_H^3 & w_j = F^2\,\mathbf{R}_U & w_p = F^2\,\mathbf{R}_H & w_v = F^2\,\mathbf{R}_H \\ w_e = F^2 & w_k = F\,\mathbf{R}_H & w_q = F^2\,\mathbf{R}_H^3 & w_w = F^2 \\ w_f = F^2\,\mathbf{R}_U & w_l = F\,\mathbf{R}_H^3 & w_r = F^2 & w_x = F\,\mathbf{R}_U \end{array}$$

3.2 Geometrical Interpretation of a P System Configuration

The structure of membranes in a cell-like P system is a tree-like graph. Such graph does not have an intrinsic geometric interpretation, but we can add such interpretation by giving a geometric meaning to the objects placed inside the membranes[2]. Given a P system configuration, the membrane structure and the

[2] These ideas has been previously used in membrane computing, see [13,21,23]. Different approaches bridging membrane computing with other geometric problems can be found in [3,4] or [16].

multisets placed in the membranes encoded all the needed information for settling the features of the modules in the robot and their relative position. Nonetheless, for a methodological point of view, such information must be interpreted in order to have a 3D model of the robot. A simple way for graphically representing a membrane structure is to make a depth-first search of it and, for each membrane containing the object F, drawing a segment of length $m \times l$, where m is the multiplicity of F and l is a length unit. This segment is drawn rotated with respect to the segment corresponding to the parent membrane with an angle of $n \times \delta$, where n is the multiplicity of objects $\mathbf{R}_i$, $i \in \{H, L, U\}$ and δ is a fixed angle ($\pi/2$ in our example). Obtaining a 3D model from a P system configuration can be made by using different methods. In this paper, the well-known turtle interpretation [1,18] is considered: A turtle is placed on an N-dimensional space (usually, $N \in \{2, 3\}$) facing in a certain direction. The turtle can move and its movements are determined by a simple object language. In its basic version the symbols only control the number of straightforward steps and the angle and direction of turns. In this way, turtle interpretation is an appropriate tool for obtaining a 3D model of a P system configuration.

4 Cluster-Flow Locomotion of Spare Modules

In this paper, the final configuration is obtained by a cluster-flow locomotion of the modules of the robot[3]. These modules move from the initial configuration to the final one determined by a P system configuration. A segment of the robot determined by a membrane containing n objects F will be built by n modules in a row. Modules have three kinds of states during the interpretation.

- **Turtle Module:** Modules which do the search work for moving as the turtle.
- **Spare Module:** Modules that can move to other areas to continue the growth of structures.
- **Finalized Module:** Modules which have reached the final position will and do not move any more.

Turtle modules do the turtle search work according to inner objects in membrane configuration, as shown in Fig. 5. Connected modules at the moving direction receive a multiset of objects in membrane configuration from former turtle modules and become new turtle modules. New turtle modules receive P system objects by reducing one F. When all neighbouring lattices meet the membrane configuration description, the turtle module changes to finalized module as a fixed part of the reconfiguration result. This locomotion allows the robot to reach the final configuration for totally connected robots. A local localization strategy is used in this decentralized control mechanism. In decentralized robotic system, there is no compass direction for turtle moving as the graphic interpretation. Instead of global map for each module, the relative orientation

[3] A detailed description of the cluster-flow locomotion is out of the scope of this paper. A good introduction can be found, e.g., in [8] or [11].

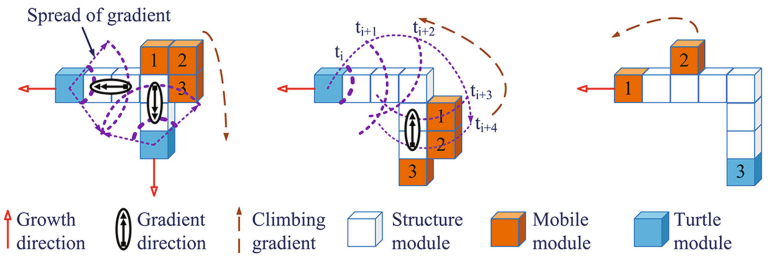

Fig. 5. Robotic segments develop by attaching new modules at neighboring lattice position in develop direction.

to connected neighbours is used. New attached modules get their global state from connected father module, which is the former turtle module. Directly connected modules can determine relative orientation through local communication on connecting surface. The regular organization of lattice-based module contributes to the computation of moving direction. The forward direction for F starts from the connecting surface receiving turtle state to the opposite surface. Turtles move by attaching a module in the neighboring lattice at the moving direction. If the neighboring lattice is not filled, spare modules in the system can move on the surface of other modules to fill it. Such modules can move on the surface of other modules, including finalized modules. Segments develop through constantly attaching new modules. CA is used for the cluster-flow locomotion of spare modules as shown in [35]. In order to get a computational model for controlling the movement of the modules, a set of CA rules has been designed, which only contains two rules. This set of rules is obviously simpler in numbers than the presented in [8]. Figure 6 show a scheme of both rules. The scheme on the left will be used for representing the convex and concave motion of one modules. The scheme on the right represents a one length movement of a module in linear motion. The gradient attraction strategy [24,35] is used to provide moving directions for local modules[4].

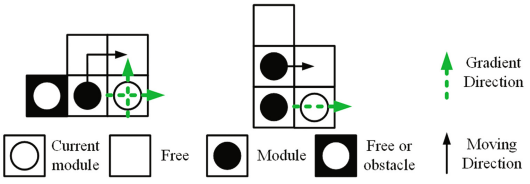

Fig. 6. A simplified set of CA rules for SCM

[4] See [35] for the technical details on the strategy for the maintenance of the global connection during the locomotion.

5 Simulation and Experimental Results

5.1 Convergence and Parallelism

Some multi-simulations have been performed in order to illustrate the used method. The membrane structure in Fig. 4(left) has been used in order to guide the self-reconfiguration process. Figure 7 shows multi simulations with increasing number of modules. All simulations start from a lattice structure and reconfigure to the predefined branching topology shown in Fig. 2. In order to verify the convergence of self-reconfiguration process, the number of modules is fixed according to the target configuration. Let us remark that decentralized modules are replaceable with each other, and there is no planning for particular position for each module. The whole structure is determined by the relative position of inner modules. Video attachments [36] record the corresponding self-reconfiguration process. The convergence of decentralized method is defined as the emergence of target structure, and has long been an open problem [14,24]. It is computed as the ratio of modules in the structure state and total modules that the target configuration. Since the robotic systems contain exactly the number of modules for target configuration, the convergence ratio increases to 100% when all modules change state to structure module. The used method is convergent in from the theoretical side, as the serial movement of turtle state between directly connected modules, but the convergence has also been verified in simulations, as shown in Fig. 8. Simulations in Fig. 7 are repeated 100 times, and simulation results are statistically analyzed. Statistical analysis shows

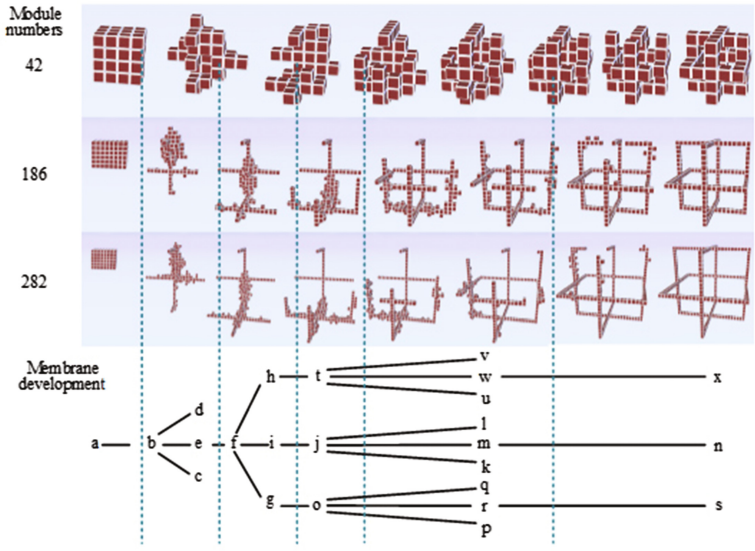

Fig. 7. Simulations with increasing number of modules by the membrane structure in Fig. 4(left).

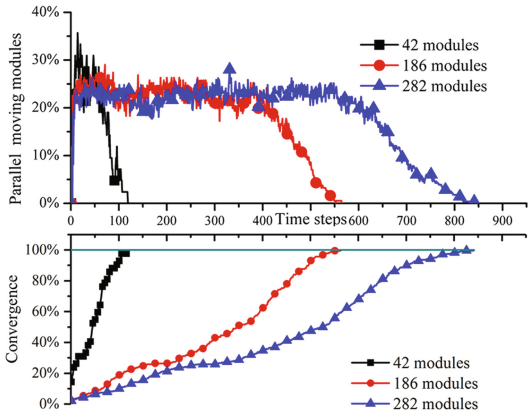

Fig. 8. Results analysis of convergent process and parallel character in decentralized modules.

the convergence of self-reconfiguration by the proposed method in Fig. 8. Compared with those emergent process [25], self-reconfigurations by the proposed method is convergent and has no unpractical assumptions. Some advantages of the physical application on real robots, and experiments are provided below. Self-reconfiguration by the proposed method is parallel in the level of independent modules. The introduction of membrane computing cooperates with the parallel nature of distributed modules. Figure 8 shows that the variation tendency in different simulations turns out to be similar. Only times steps for global self-reconfiguration increases along the increase of module numbers, moving modules in parallel in each simulation maintains the same tendency.

5.2 Scalability

The designed method is scalable to module numbers. The scalability of control method cooperates with the mechanical scalability by modular design of MSR robots. Multi simulations have been performed and the time steps for global convergence is shown in Fig. 9. Compared with the exponential increase of control complexity in centralized control, time steps for self-reconfiguration using the proposed method in this paper increases linearly with the number of modules.

Though modules can perform motion planning and move independently and simultaneously, the development style of robotic structure during self-reconfiguration needs to attach modules one by one on the growing front. One influencing factor of the increasing time steps lies in the relative attachment of decentralized modules for the development and reconfiguration of MSR robots. Another influencing factor lies in the movement of modules by climbing gradient along robotic segments.

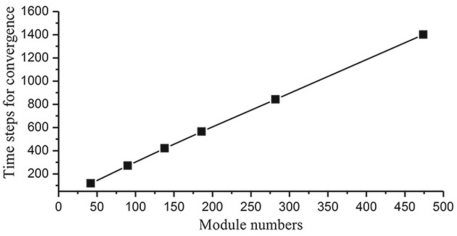

Fig. 9. Scalability to module numbers.

5.3 Experiments

The proposed method has also been verified on Modular *Se*lf-*re*configurable *Mo*bile (Seremo) robots which have been developed in the hardware laboratory at the Harbin Institute of Technology (Fig. 10(Left)). With a 80×80 (mm) lattice structure in 2D, a Seremo module is equipped with local communicating ability and local sensing ability on four connecting faces. Each module is a complete robot by itself with *onboard* sensors, actuators, processor, battery and means of communication[5]. The Seremo modules can move by rotating along the edge of neighboring modules. As shown in Fig. 10(Right), Seremo modules can achieve the convex, concave and linear motion of lattice motion in 2D. Seremo modules can autonomously connect to and disconnect from neighboring modules. Robots with Seremo modules can achieve decentralized locomotion by repeating the basic motion in Fig. 10(Right). A simple experiment with a membrane structure with two branches is shown in Fig. 11. Each branch is a segment with s modules linearly connected. All initial configurations starts from a linear structure and reconfigurations start from the first module in the right. Some experiments (with $s = 1, 2, 3, 4$) show the convergence to the topology defined by the membrane structure. Obstacles are also placed in experiments with $s = 3, 4$. With local sensing of Seremo modules, neighboring modules can sense the existing of obstacles and translate through local communication. The surface locomotion of Seremo modules can move around to the growing front according to local information. Video attachments [36] record the corresponding self-reconfiguration process.

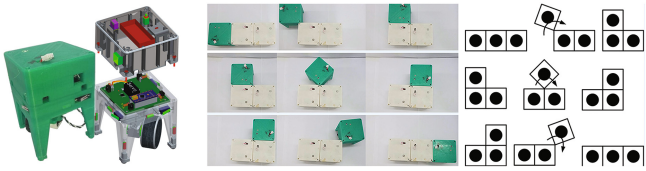

Fig. 10. Seremo robots (Left). Lattice motion of Seremo modules (right).

[5] More details about the mechanical and electrical structure can be found at [6].

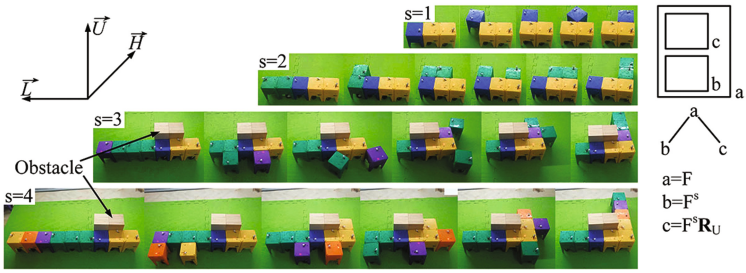

Fig. 11. Some experiments with Seremo robots.

6 Conclusions

The control of self-reconfigurable robots in an extremely hard task. On the one hand, it needs the development of efficient hardware modules able of performing quick and precise movements in two or three dimensions, but, on the other hand, it needs of theoretical contributions which guide the movements efficiently. Such moves depend on several changing variables. Each module processes information locally and independently from the other modules. Several modules can move simultaneously in order to reach the final configuration. Such configuration is not determined by a global position in a 3D space, but it is only determined by the relative position among the modules. In order to find an appropriate control of decentralized modules, many different ideas coming from different research areas are brought together in this paper. Among them, the partition of the Euclidean space into similar tiles where a module can be *alive* or not; a turtle interpretation for giving a *dynamic* meaning to a set of *static* symbols representing a configuration or a gradient attraction strategy for moving independent modules. The use of ideas from membrane computing can shed a new light to the local representation of the information. As pointed out above, the approach used in this paper can be considered from both theoretical and practical side, enriching each other with ideas coming from the self-configurable robots and from membrane computing. Such open research lines involve new developments in the application of the theoretical framework of membrane computing for the abstract representation of robots (and hence a deeper understanding of the theoretical possibilities) and also, from the practical side, the development of new abilities of physical robots inspired in the local encapsulation of the information. Many problems remain open. The study of how problems and techniques from both research areas can provide new solutions on the other side is matter of future research.

Acknowledgement. This work was supported by National Natural Science Foundation of China (Grant No. 61673137) and the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 51521003).

References

1. Abelson, H., DiSessa, A.A.: Turtle Geometry: The Computer as a Medium for Exploring Mathematics. MIT Press, Cambridge (1986)
2. Baca, J., Woosley, B., Dasgupta, P., Nelson, C.A.: Configuration discovery of modular self-reconfigurable robots: real-time, distributed, Ir+XbEe communication method. Robot. Auton. Syst. **91**, 284–298 (2017)
3. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G.: Simulation of Spatial P System Models. Theor. Comput. Sci. **529**, 11–45 (2014)
4. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G., Tesei, L.: Spatial P systems. Nat. Comput. **10**(1), 3–16 (2011)
5. Bie, D., Gutiérrez-Naranjo, M.A., Zhao, J., Zhu, Y.: A Membrane Computing Framework for Self-Reconfigurable Robots (submitted)
6. Bie, D., Zhu, Y., Wang, X., Zhang, Y., Zhao, J.: L-systems driven self-reconfiguration of modular robots. Int. J. Adv. Robot. Syst. **13**(5), 1–12 (2016)
7. Bojinov, H., Casal, A., Hogg, T.: Multiagent control of self-reconfigurable robots. Artif. Intell. **142**(2), 99–120 (2002)
8. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic decentralized control for a class of self-reconfigurable robots. In: IEEE International Conference on Robotics and Automation, ICRA 2002, vol. 1, pp. 809–816 (2002)
9. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Cellular automata for decentralized control of self-reconfigurable robots. In: IEEE ICRA Workshop on Modular Robots, pp. 21–26 (2001)
10. Butler, Z.J., Kotay, K., Rus, D., Tomita, K.: Generic decentralized control for lattice-based self-reconfigurable robots. Int. J. Robot. Res. **23**(9), 919–937 (2004)
11. Fitch, R., Butler, Z.J.: Scalable locomotion for large self-reconfiguring robots. In: IEEE International Conference on Robotics and Automation, pp. 2248–2253 (2007)
12. Fukuda, T., Nakagawa, S.: Approach to the dynamically reconfigurable robotic system. J. Intell. Robot. Syst. **1**(1), 55–72 (1988)
13. Georgiou, A., Gheorghe, M., Bernardini, F.: Membrane-based devices used in computer graphics. In: Ciobanu, G., Păun, G., Pérez-Jiménez, M.J. (eds.) Applications of Membrane Computing. Natural Computing Series, pp. 253–281. Springer, Heidelberg (2006)
14. Hou, F., Shen, W.: Graph-based optimal reconfiguration planning for self-reconfigurable robots. Robot. Auton. Syst. **62**(7), 1047–1059 (2014)
15. Lakhlef, H., Bourgeois, J., Mabed, H., Goldstein, S.C.: Energy-aware parallel self-reconfiguration for chains microrobot networks. J. Parallel Distrib. Comput. **75**, 67–80 (2015)
16. Margenstern, M.: Can hyperbolic geometry be of help for P systems? In: Martín-Vide, C., Mauri, G., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2003. LNCS, vol. 2933, pp. 240–249. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24619-0_18
17. Perez-Diaz, F., Zillmer, R., Groß, R.: Control of synchronization regimes in networks of mobile interacting agents. Phys. Rev. Appl. **7**, 054002 (2017)
18. Prusinkiewicz, P.: Graphical applications of L-systems. In: Proceedings on Graphics Interface 1986/Vision Interface 1986, pp. 247–253. Canadian Information Processing Society, Toronto (1986)
19. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Virtual Laboratory. Springer, New York (1990)

20. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford (2010)
21. Rivero-Gil, E., Gutiérrez-Naranjo, M.A., Romero Jiménez, Á., Riscos-Núñez, A.: A software tool for generating graphics by means of P systems. Nat. Comput. **10**(2), 879–890 (2011)
22. Romanishin, J.W., Gilpin, K., Claici, S., Rus, D.: 3D M-blocks: self-reconfiguring robots capable of locomotion via pivoting in three dimensions. In: IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26–30 May 2015, pp. 1925–1932. IEEE (2015)
23. Romero-Jiménez, A., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Graphical modeling of higher plants using P systems. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2006. LNCS, vol. 4361, pp. 496–506. Springer, Heidelberg (2006). https://doi.org/10.1007/11963516_31
24. Stoy, K.: Using cellular automata and gradients to control self-reconfiguration. Robot. Auton. Syst. **54**(2), 135–141 (2006)
25. Stoy, K.: Lattice automata for control of self-reconfigurable robots. In: Sirakoulis, G.C., Adamatzky, A. (eds.) Robots and Lattice Automata. ECC, vol. 13, pp. 33–45. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-10924-4_2
26. Stoy, K., Brandt, D., Christensen, D.J.: Self-Reconfigurable Robots: An Introduction. Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge (2010)
27. Valentini, G., Ferrante, E., Hamann, H., Dorigo, M.: Collective decision with 100 kilobots: speed versus accuracy in binary discrimination problems. Auton. Agent. Multi-Ag. **30**(3), 553–580 (2016)
28. Vergara, A., Lau, Y.S., Mendoza-Garcia, R.F., Zagal, J.C.: Soft modular robotic cubes: toward replicating morphogenetic movements of the embryo. Plos One **12**(1), 1–17 (2017)
29. Wang, X., Jin, H., Zhu, Y., Chen, B., Bie, D., Zhang, Y., Zhao, J.: Serpenoid polygonal rolling for chain-type modular robots: a study of modeling, pattern switching and application. Robot. Cim-Int. Manuf. **39**, 56–67 (2016)
30. Wolfram, S.: Cellular Automata and Complexity: Collected Papers. Addison-Wesley, Reading (1994)
31. Yang, Z., Wu, Y., Fu, Z., Fei, J., Zheng, H.: A unit-compressible modular robotic system and its self-configuration strategy using meta-module. Robot. Cim-Int. Manuf. **49**, 39–53 (2018)
32. Yim, M., Shen, W., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robot. Autom. Mag. **14**(1), 43–52 (2007)
33. Zhang, Y., Song, G., Liu, S., Qiao, G., Zhang, J., Sun, H.: A modular self-reconfigurable robot with enhanced locomotion performances: design, modeling, simulations, and experiments. J. Intell. Robot. Syst. **81**(3–4), 377–393 (2016)
34. Zhao, J., Wang, X., Jin, H., Bie, D., Zhu, Y.: Automatic locomotion generation for a ubot modular robot-towards both high-speed and multiple patterns. Int. J. Adv. Robot. Syst. **12**, 32 (2015)
35. Zhu, Y., Bie, D., Iqbal, S., Wang, X., Gao, Y., Zhao, J.: A simplified approach to realize cellular automata for ubot modular self-reconfigurable robots. J. Intell. Robot. Syst. **79**(1), 37–54 (2015)
36. https://sites.google.com/site/modularrobots/cycling