A Simple and Strong Algorithm for Reconfiguration of Hexagonal Metamorphic Robots

KwangEui Lee[†]

[†] Department of Multimedia Engineering, Dongeui University, Busan, Korea

Summary
In this paper, we propose a reconfiguration algorithm for planar hexagonal metamorphic robot systems. We consider a distributed reconfiguration algorithm that doesn't need any preprocessing or global communications. Additionally, our algorithm requires only little communication among adjacent modules. Our algorithm has no constraints on initial and goal configurations and no preprocessing steps. For this algorithm, we propose a new classification method for neighbor contact patterns which is easy to extend for various neighbor groups.

Key words:
Hexagonal metamorphic robot, Distributed Reconfiguration, Self-reconfigurable robot system.

1. Introduction

A metamorphic robot system is a collection of independently controlled homogeneous mobile robots [1], each of which has the same shape, the same ability to connect and disconnect to other modules. So they are completely interchangeable. This interchangeability provides a high degree of system fault tolerance. In metamorphic systems, modules autonomously change their relative position in order to change the shape of the overall system. This process called self-reconfiguration [2].
Potential applications of metamorphic systems include obstacle avoidance in highly constrained and unstructured environments, 'growing' structures composed of modules to form bridges, buttresses, and other civil structures in times of emergency [3].
In this paper, we propose a distributed reconfiguration algorithm for metamorphic robot systems. We focus on a system of planar, hexagonal robotic modules. We consider a distributed motion planning algorithm that doesn't need any preprocessing or global communications. Additionally, our algorithm requires much fewer communications among modules compared to other approaches.
This paper has three main contributions: 1) it provides a very simple and strong reconfiguration algorithm for general shape of initial and goal configurations, 2) there are no preprocessing steps, and 3) we suggest a new classification method for neighbor contact patterns. The rest of the paper is organized as follows. We introduce some previous works in section 2. In section 3 and 4, we present the system model and our algorithm for reconfigurations, respectively. The analysis and experimental results are shown in section 5. Finally, section 6 draws the conclusion.

Manuscript received February 5, 2010
Manuscript revised February 20, 2010

2. Related Works

The motion planning for a metamorphic robotic system is to determine the sequence of robot motions required to reconfigure the whole system to a desired goal configuration from a given initial configuration.
Unfortunately, there is no simple method for solving the motion planning problem. Because, given any number of modules, the number of possible connected configurations is exponential to the number of the modules [3]. So, there are many researches based on heuristic methods or restricted configurations in the literatures.
Many existing motion planning algorithms use centralized method to plan and supervise the robot motion [2], [3], [4]. Some others propose distributed approaches which based on heuristic approximations and require communication among robots in the reconfiguration process [5], [6]. And some others suggest distributed deterministic algorithms with preprocessing [7], [8], [9], [10].
Proposed algorithm is a distributed deterministic algorithm. Comparing to previous algorithm, proposed algorithm takes little bit more step to reconfigure the system. But the proposed algorithm is still valuable in the point of view that there is no preprocessing or no global communications. Above all it is very simple and strong.

3. The System Model and Some Definitions

3.1 Coordinate System

We consider hexagonal metamorphic robot system. Each module has identical hexagon shape and occupies exactly one cell in the plane. We assume that the plane is partitioned and labeled as shown figure 1. It is the same coordinate system described in [1]. The coordinate system is generated from the patch shown in figure 2. In figure 2, N, S, W, E means North, South, West, and East respectively.

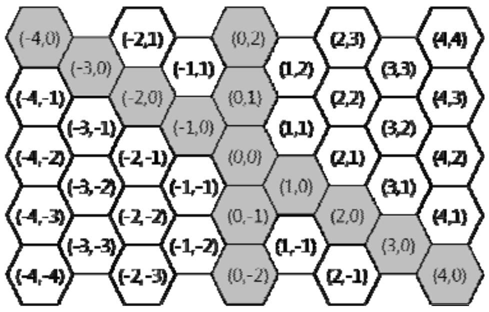


Fig. 1. Coordinate system.

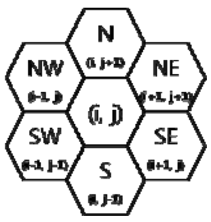


Fig. 2. Numbering convention.

3.2 Module Types and Assumptions

Each module runs the same program and moves by rotating around another module, called substrate, clockwise (CW) or counter clockwise (CCW). There are three types of modules:

1. Base: immovable base module, every other module should be connected to the base directly or in directly via their neighbors.
2. Robot: the module explained above.
3. Obstacle: kind of a robot but it can't move.

In our algorithm, there are no global coordinators or global communication among modules. Modules need only local communication with their neighbors and some initial knowledge about the position of the goal cells. At all times, each module knows its location and its unique robot identifier (RID).

Modules move in lockstep rounds. In each rounds, a module M (in figure 3) can move clockwise (counter clockwise) if the cell S does not move, and cell C2 and C3 (C1 and C2) are empty.

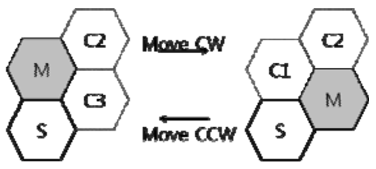


Fig. 3. Moving a module in the plane.

3.3 Module Contact Patterns

In [1], they classify the module contact patterns in 12 categories as shown in figure 4.

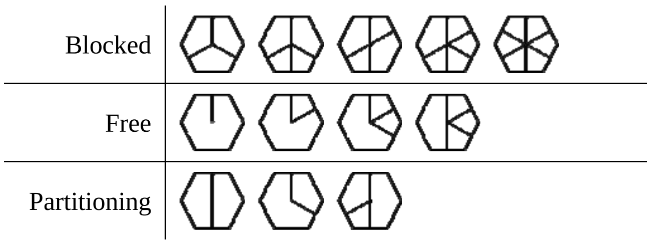


Fig. 4. Possible contact patterns.

Our algorithm also uses these contact patterns of neighbor, but at the same time our algorithm sees these patterns as a 6 bit integer. For example, the connection pattern for the gray module in figure 5 will be represented as 001101. In the pattern, '0' means the absence of neighbor and '1' means the presence of neighbor in the order of 'N', 'NE', 'SE', 'S', 'SW', 'NW'. So, there are 64 patterns from 000000 to 111111 and 0 to 63 in the decimal system.

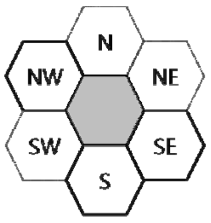


Fig. 5. A sample neighbor pattern.

The bit patterns are also classified into 13 contact patterns, but every bit pattern is considered separately in our algorithm. We add mirror image of the last pattern of 'Partitioning' patterns of figure 4. To consider the direction of modules destination, we treat every bit pattern separately even they classified as a same contact pattern. But the contact pattern is still very useful information in our algorithm.

4. Algorithm for Reconfigurations

The overall algorithm runs in 3 steps:

1. Decide the moving direction of each module.
2. Check the substrate to prevent unpredictable movement.
3. Check the neighbors to maintain the connectivity.

In the following we will show the details of each step.

4.1 Deciding the Moving Direction

There are 64 contact patterns in 13 categories. But, for now, we use only 18 patterns in 3 categories for simplicity. This will be extended in further researches. These 3 categories are the first 3 contact patterns in FREE categories of figure 4. Table 1 shows the moving direction for each pattern.

Table 1: Moving direction for each pattern.

pattern	direction	pattern	direction
000001(1)	N	011000(24)	S
000010(2)	NW	110000(48)	SE
000100(4)	SW	100001(33)	NE
001000(8)	S	000111(7)	N
010000(16)	SE	001110(14)	NW
100000(32)	NE	011100(28)	SW
000011(3)	N	111000(56)	S
000110(6)	NW	110001(49)	SE
001100(12)	SW	100011(35)	NE

Algorithm 1 shows how to generate the pattern from neighbor contact information. We use pseudo-code for brevity.

```
// algorithm 1: pattern_generation

enum direction {N=0,  NE, SE, S, SW, NW };
pattern =0;
for (int i=0; i<6; i++) {
        pattern *=2;
        if (neighborInDirection[i] is  a ROBOT)
            pattern +=1;
}
```

4.2 Checking the Substrate

If the algorithm does not ensure immobile substrates, the results of the round are unpredictable. So, we check the substrate first. If the substrate is moving, then the module cancels the move. Each module can communicate with their neighbors so this process can be easily implemented using local communications.

4.3 Checking the Neighbors

Because of the algorithm uses only 16 patterns and modules are always move CCW, there is only one possibility to break the connectivity after a round. Figure 6 shows the case.

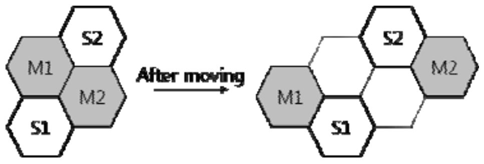


Fig. 6. The only connection pattern will be disconnected.

To prevent disconnection, before move, we check the neighbors' id. Only the module with the lowest RID has the right to move. Algorithm 2 shows how to implement the process.

```
// algorithm 2: check_neighbor

enum direction {N=0,  NE, SE, S, SW, NW };
lowestID = 0;
for (int i=0; i<6; i++)
        if ((neighborInDirection[i]==ROBOT)
        && (neighborInDirection[i].rid< lowestID))
            lowestID = neighborInDirection[i].rid;
if (myrid< lowestID) move;  else do not move;
```

5. Experimental Results

We developed a java based object-oriented simulator to test our reconfiguration algorithms. In this section we will show the experimental results.

5.1 Algorithm Demonstration

Figure 7 shows initial configuration which will be solved by our algorithm. In the Figure, B, O, R, G, E means Base, Obstacle, Robot, Goal cell, Empty cell respectively.

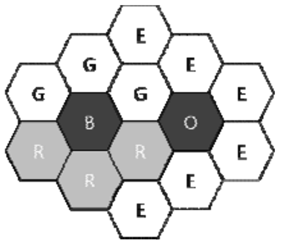


Fig. 7. Initial configuration.

In figure 8, we show the process of changing configurations by the proposed algorithm.

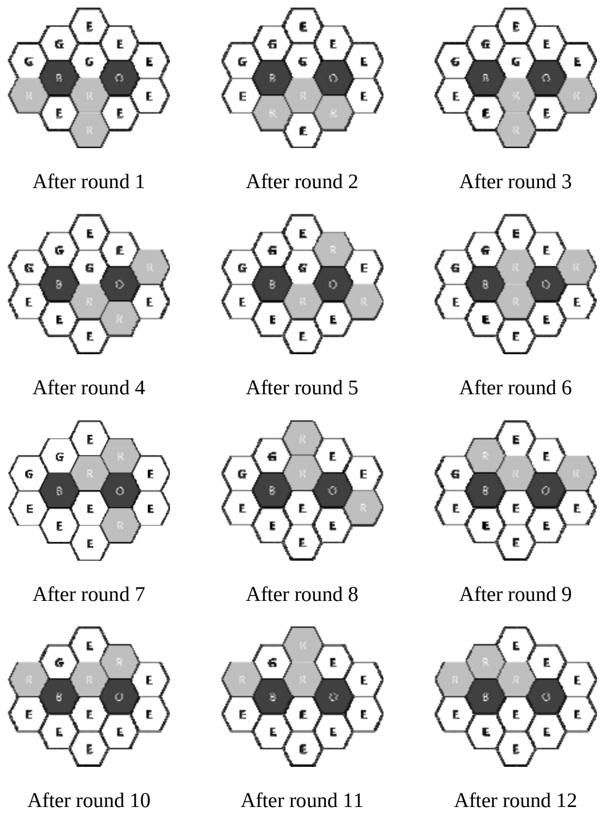


Fig. 8. Process of changing configurations.

5.2 Comparison to Previous Results

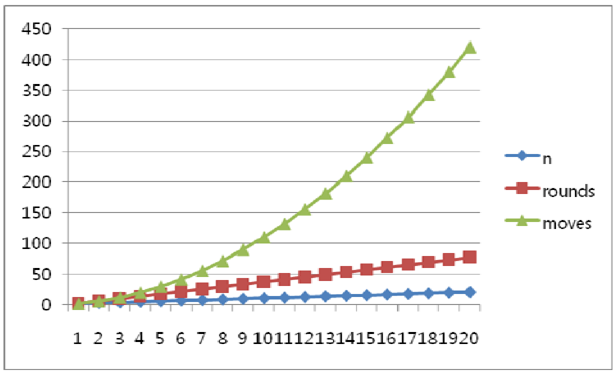
Here, we compare the result of proposed algorithm to the result of J. Walter et al. [5]. As the result shows, proposed algorithm runs 141% more rounds, but, the proposed algorithm is still valuable in the point of view that there is no preprocessing, and no constrains on the initial configurations in the algorithm.

Table 2: Comparison to the result of J. Walter et al.

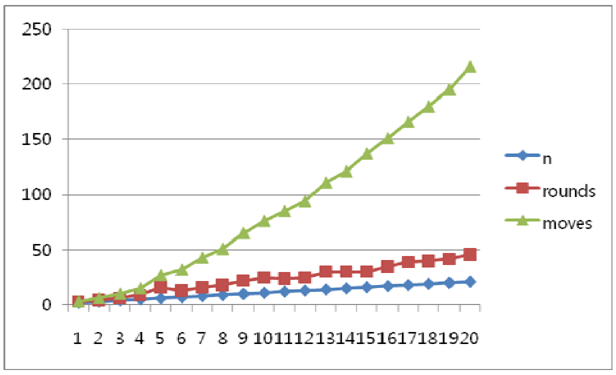
Size(n)	Walter et al.	Ours	O/W ratio
11	24	34	142%
16	35	46	131%
10	22	28	127%
37	103	123	119%
35	77	134	174%
24	58	84	145%
total	319	449	141%

5.3 Experimental Results on Various Configurations

Finally, we test the relationship between the number of modules and rounds (and the number of module moves). We carried out two types of experiments. The first one is the case that both initial and goal configurations are linear. In the second experiment, both initial and goal configurations are square. Figure 9 shows the result.



(a) Linear initial configuration and linear goal configuration



(b) Square initial configuration and square goal configuration

Fig. 9. Running time in rounds.

6. Conclusions and Further Works

In this paper, we propose a distributed reconfiguration algorithm for planar hexagonal metamorphic robot system. Proposed algorithm doesn't need any preprocessing or global communications and requires significantly fewer communications among modules. Also, we propose a new classification method for neighbor contact patterns.

Further studies may include extending the algorithm in 3D case and adding global coordinators that will be the skeleton of the goal configuration. We think, these global coordinators are essential to rapid reconfiguration for very complex goal configuration.

Acknowledgments

This work was supported by Dong-eui University Foundation Grant (2007)

References

[1] G. S. Chirikjian, “Kinematics of a metamorphic robotic system,” in Proc. of IEEE Intl. Conf. on Robotics and Automation, pp. 449-455, 1994.
[2] K. Kotay, D. Rus, M. Vona, and C. McGray, “The self-reconfiguring robotic molecule: Design and control algorithms,” in Proc. Workshop Algorithmic Foundations of Robotics, pp. 376–386, 1998.
[3] A. Pamecha, I. Ebert-Uphoff, and G. Chirikjian, “Useful metrics for modular robot motion planning,” in IEEE Transactions on Robotics and Automation, vol. 13, pp. 531-515, 1997.
[4] A. Casal and M. Yim, “Self-reconfiguration planning for a class of modular robots,” in Proc. SPIE Symp. Intelligent Systems and Advanced Manufacturing, vol. 3839, pp. 246–256, 1999.
[5] S. Murata, H. Kurokawa, and S. Kokaji, “Self-assembling machine,” in Proc. IEEE Int. Conf. Robotics and Automation, pp. 441–448, 1994.
[6] Y. Zhang, M. Yim, J. Lamping, and E. Mao, “Distributed control for 3D shape metamorphosis,” Autonomous Robots, vol. 10, no. 1, pp. 41–56, Jan. 2001.
[7] J. Walter, E. Tsai, and N. Amato, “Choosing good paths for distributed reconfiguration of hexagonal metamorphic robots,” in Proc. IEEE Int. Conf. Robotics and Automation, pp. 102–109, 2002.
[8] J. Walter, J. Welch, and N. Amato, “Distributed reconfiguration of metamorphic robot chains,” in Proc. ACM Symp. Principles of Distributed Computing, pp. 171–180, 2000.
[9] J. Walter, E. Tsai, and N. Amato, "Algorithms for Fast Concurrent Reconfiguration of Hexagonal Metamorphic Robots," IEEE TRANSACTIONS ON ROBOTICS, VOL. 21, NO. 4, pp. 621-631, 2005.
[10] S. Matysik, J. Walter, "Using a Pocket-Filling Strategy for Distributed Reconfiguration of a System of Hexagonal Metamorphic Robots in an Obstacle-Cluttered Environment," in Proc. of IEEE Intl. Conf. on Robotics and Automation, pp. 4265-4272, 2009.

KwangEui Lee received his B.S., M.S. and Ph.D. degrees from Sogang University, Seoul, Korea in 1990, 1992, and 1997, respectively. From 1997 to 2001, he joined ETRI as a senior research member. Since 2001, He has been an associate professor of Dongeui University. His research interests include computation theory, artificial life and their applications.