Detector: Hierarchical Distributed Fault Detection Algorithm for Lattice Based Modular Robots

Edy Hourany, Benoit Piranda, Abdallah Makhoul, Julien Bourgeois, Bachir Habib

Abstract Modular robots are robots built from multiple modules which can reconfigure the shape of the whole robot. This makes them a promising technology to realize highly adaptable and re-configurable robots. One of the main challenges in modular robotics is to detect and heal the system from a state of partial or complete disconnection. Partial disconnection can be achieved by removing the links between the modules. Complete disconnection can be achieved by removing all the connections between two group of modules. It is important for the modular robots to detect a failure between connections since the mobility and the functionality of the robot depend on it. This paper presents a novel approach to detect disconnections in lattice based modular robots. It detects the disconnections using a hierarchical algorithm. The leader of the system constructs a spanning tree, assigns a coordinate to each module and compares it to detect the disconnections. The simulations shows the ability to detect all partial disconnections. Furthermore, the complexity regarding the number of messages sent is linear against the number of modules in the system.

1 Introduction

The concept of programmable matter was first introduced by Christopher Langton in 1986 [9]. The term, "Programmable Matter" itself, was first introduced by Tommaso Toffoli and Norman Margolus [17] and has also been used in the context of artificial life, where it is referred to as digital organisms or artificial life.

Programmable matter is not a single robot or machine, but rather a network of all of the above [2]. It is a fabric that can change its physical properties in real

Edy Hourany, Benoit Piranda, Abdallah Makhoul, Julien Bourgeois
FEMTO-ST institute, Univ. Bourgogne Franche-Comté, CNRS, 1 cours Leprince-Ringuet, 25200, Montbéliard, France, e-mail: {first}.{last}@femto-st.fr

Bachir Habib
Holy Spirit University of Kaslik, Jounieh, Lebanon 446 e-mail: bachirhabib@usek.edu.lb

time [11]. Programmable matter can respond to external or internal cues, or can be triggered by the completion of a task. It can grow and adapt, and self-assemble and self-repair [6, 18].

Programmable matter consists of a group of micro-robots, called granular computing elements. Each of these robots can move in space, sense the environment and communicate with each other. Each robot can communicate with its connected neighbor to exchange information about its own location and surroundings, and perform the computation in parallel. The robots are small, and can be manufactured in a highly parallel and distributed fashion. The difference between these robots and a traditional micro-robot is that these robots self-organize into a structure that is not readily predictable, and changes in response to a variety of environmental cues [18].

The benefit of programmable matter is that its capabilities are not limited to any particular design or configuration. The robots can be configured in a variety of shapes, sizes, and locations [6, 11, 18]. The robots can be added, removed or reorganized without having to re-design and re-manufacture the structure. The robots can even be built from other robots, or robots made from other materials. A programmable matter is more than a collection of objects with the ability to move and change shape.

A very frequent error in modular robots is disconnection. It can be caused by incorrect design, lack of care, or a defective module. Disconnections among robots can lead to collisions or a lack of mobility, which will make the robot unable to perform its function. Disconnections can also cause a loss of autonomy or even the robot's death. Therefore, it is crucial to detect disconnection among modular robots to prevent errors in the overall system. This paper proposes a novel hierarchical disconnection detection algorithm based on spanning tree that detects all partial disconnections in a lattice based modular robotic system.

The remainder of this paper is as follow: Section 2 presents the literature review, Section 3 presents the proposed algorithm, Section 4 shows the simulation results, and finally Section 5 concludes the paper.

2 Background

Modular robots are mobile robots built from multiple units, usually called modules or particles, capable of interactions with each other. These units are connected by links, which are able to transmit forces between the units. Several architecture arises from this technology such as hybrid architecture, chain architecture, and lattice architecture. Some examples of modular robots are shown in Figure 1.

In modular robotics, the mobility of the modules is often not controlled by the computer itself, but the modules communicate among each other and decide among themselves how to move. Simple tasks such as collision detection, passing obstacles and shape formation become more complex with this approach.

Disconnections among modular robots can occur for a number of reasons: (1) a robot may move off-course, (2) a robot may collide with a fixed structure, (3)

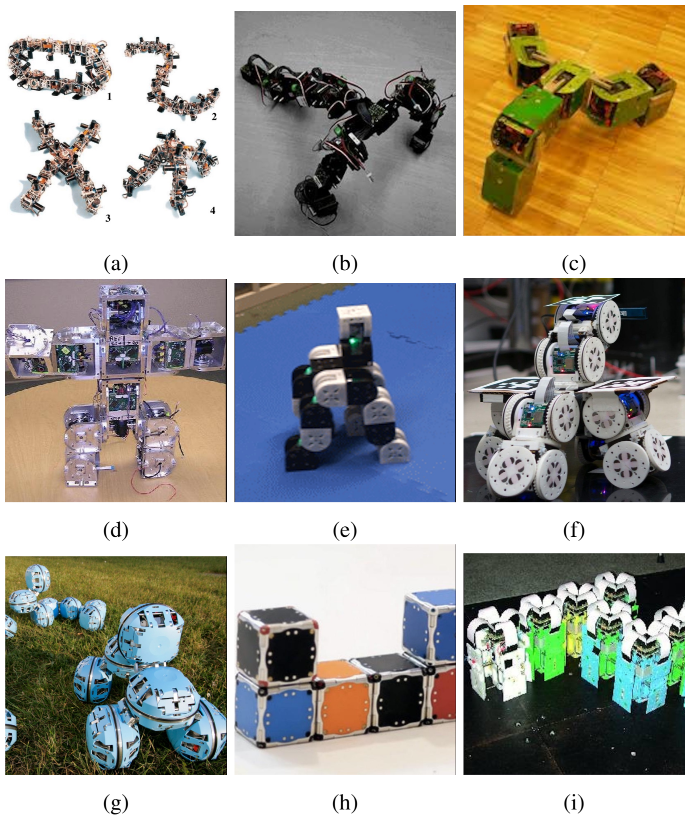


Fig. 1: Examples of modular robots: Figure 1a represents a polybot introduced in [19]; Figure 1b is a Tripod configuration of CONRO [3]; Figure 1c is a Five YaMoR modules [10]; Figure 1d is a lattice modular robot called ATRON [7]; Figure 1e is a Rotating M-Block [13]; Figure 1f is a 2D Crystalline [14] modular robots with extensible arms; Figure 1g is a Superbot [15] modular robot in a humanoid configuration. Figure 1h M-TRAN [8] modular robot in 4-legged configuration; Figure 1i is a SMORES [4] modular robot.

two robots may physically touch each other, (4) two robots may physically touch each other and then become disassociated from the system, and/or (5) a robot may disconnect from the system through no fault of its own. While a modular robotic system is being operated, the system must ensure that all communications remains available. In order to achieve this, it is crucial to detect disconnections among the robots of the system. Such disconnections may, for example, result in the loss of valuable data that is being transferred among the robots, or in the loss of robot operation time.

A modular robotic system can be represented as an undirected graph, but traditional graph algorithms, such as Weakly Connected Components [5], can't be applied due to the system's distributed nature.

In [20], the authors presented a machine learning based algorithm to detect faults in distributed robotics system. Their work can be summarized as follows: Creation of data-sets of multi-robot system with faults and failures for the purpose of learning faults.

In [12], Fault detection over a wireless sensor network in a fully distributed manner. First, they proposed the Convex hull algorithm to calculate a set of extreme points with the neighbouring nodes and the duration of the message remains restricted as the number of nodes increases. Then they proposed a Naïve Bayes classifier and convolution neural network (CNN) to improve the convergence performance and find the node faults.

In [1] A distributed fault detection scheme for modular and reconfigurable robots (MRRs) with joint torque sensing was proposed. With the proposed scheme, the joint torque command is filtered and compared with a filtered torque estimate derived from the nonlinear dynamic model of MRR with joint torque sensing. Common joint actuator faults are considered with fault detection being performed independently for each joint module.

Modular robotics systems have limited memory and processing resources, a decentralized configuration, and a distributed architecture. Therefore, despite their accuracy and effectiveness, the previously described algorithms will not work.

This paper will be presenting a novel algorithm that detects disconnections in lattice based modular robots. The algorithm starts by constructing a spanning tree with the leader as root. Then, starting from the leaves, each module will send the list of none existing neighbors to their parents. Each parent will compare the received list with the list of the existing neighbors to detect any disconnections

3 Detector: disconnection detection

This section will introduce the proposed algorithm. Lets start by presenting some definitions and terminologies.

3.1 Definitions

Definition 1 (Leader). The leader is a module that was elected, using a leader election algorithm, before the execution of the proposed algorithm.

Definition 2 (Connected Modules). Connected modules are latched neighbors that are able to communicate throw their common interface.

Definition 3 (Partial disconnection). A partial disconnection is a set of nodes that are not connected to each other but are able to communicate through other common nodes on the system. Figures 3a and 3b shows an example of partial disconnection.

Definition 4 (Complete disconnection). A node is referred to be in a state of complete disconnection when it looses connections with all its neighbors. Making it impossible to be identified by the system and will be in a state of complete isolation. Figures 3c and 3d shows an example of complete disconnection.

Definition 5 (Lattice architecture). The units of a lattice architecture link their docking interfaces at points into virtual cells of a regular grid. This network of docking sites is analogous to atoms in a crystal, and the grid to the crystal's lattice. As a result, the kinematical characteristics of lattice robots may be defined by their corresponding crystallographic displacement groups (chiral space groups). Typically, just a few units are required to complete a reconfiguration stage. Lattice designs provide a simpler mechanical design as well as a simpler computational representation and reconfiguration planning that can be scaled more readily to complex systems.

Definition 6 (Density). The density of a modular robotic system shows how scattered are the modules. The value of the density ranges between 0 and 1. A system is considered to be dense when its density value is close to 1. The coordinates assigned to the modules does not take into consideration the size of the module. Each module is considered to have one unit size. Equation 1 shows how to compute the density in a system. N represents the total number of modules. x_{max}, y_{max}, and z_{max} depicts the maximum coordinates in the lattice and x_{min}, y_{min}, and z_{min} depicts the minimum coordinates in the lattice.

$$Density = \frac{N}{(x_{max} - x_{min} + 1) * (y_{max} - y_{min} + 1) * (z_{max} - z_{min} + 1)} \tag{1}$$

Definition 7 (Potential neighbors). A potential neighbor is defined as the not connected side of the robot. Two cases are presented, connection failure or absence of modules. Let's take module B in Figure 2 as an example. Module B in this condition has two potential neighbors: The none existing one on the right side and the disconnected one-Module D-.

3.2 The Proposed Algorithm

This section will be describing the proposed solution in details. The algorithm is based on three phases.

- Phase 1: Spanning tree creation and coordinate distribution;
- Phase 2: Neighbors' Data Collection;
- Phase 3: Disconnection detection. Below is a detailed explanation of the phases.

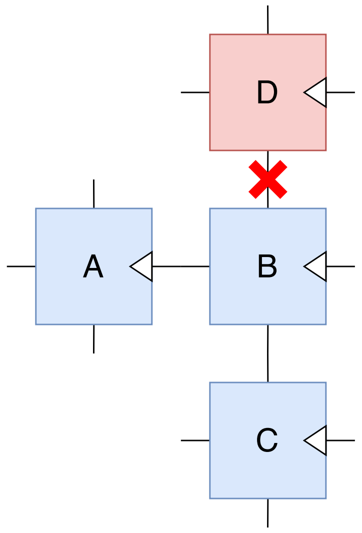


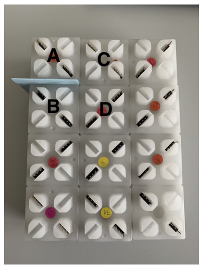
Fig. 2: Potential Neighbors

3.2.1 Spanning tree creation and coordinate distribution

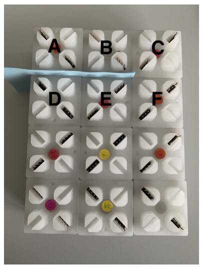
The leader of the system initiates the algorithm. It will create a spanning tree containing all the modules in the system. The process starts by sending a recruitment message to all its neighbors. Once a module receives a recruit message, it will change the recruited flag to 1, to prevent future recruitment from other modules. Mark the interface that sends the recruit message as parent interface. Extract the coordinates from the received message and update the coordinates based on the received direction in order to generate its own coordinate. And finally, send a recruit message to all its neighbor except for its parent.

3.2.2 Neighbors' Data Collection

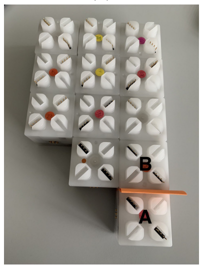
The proposed suggestion to find disconnections in a system is to gather information about the neighbours. The process starts with the leaves. Each leaf will generate the coordinates for its potential neighbors, add those coordinates into a vector - that will be called potentialNeighborsVector - and send this vector to its parent alongside their own ID and coordinates. Each parent will also generate the potential neighbors coordinates, check for possible disconnections - explained in Subsection 3.2.3 - then send the results to its parent. The same process will be repeated until the vector reaches the leader. Figure 4 shows an illustrative example of this phase.



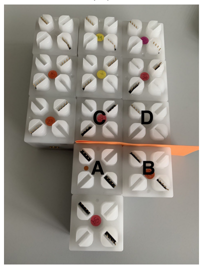
(a)



(b)



(c)



(d)

Fig. 3: Different examples of Partial and complete disconnections: Figures 3a and 3b Represents a partial disconnection. In Figure 3a. modules A and B are disconnected, but they still are part of the whole system since they can maintain their connection to the system through their neighbors C and D. Figures 3c and 3d are in a state of complete disconnection. In Figure 3c, module A is disconnected from module B and module A is not connected to any other module. Therefore, module A is considered to be in complete disconnection from the system since there is no other way for the system to acknowledge its presence. Same case for Figure 3d.

3.2.3 Disconnection detection

Once a node receives from all its children the vector potentialNeighborsVector, it will compare with the list of coordinates of existing modules. If the node finds potential neighbors coordinates matching with existing modules, then there is a disconnection. Since each node have access to only its children, then the final decision of disconnection is actually at the leader's level.

4 Simulation Results

The proposed algorithm was implemented and evaluated using VisibleSim [16] a modular robots simulator. To study the performance of the proposed algorithm, the metrics that were taken into account in the simulations are the number of messages

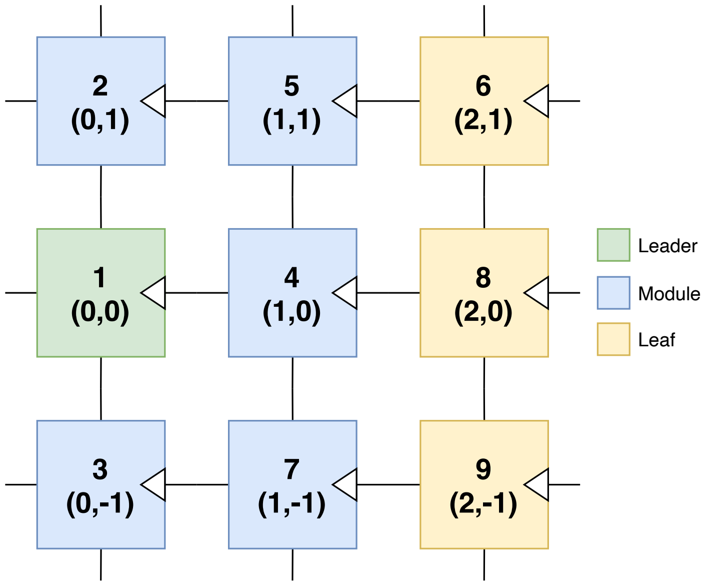


Fig. 4: Data collection in a dense system: Module 6 will add to the potentialNeighborsVector the following coordinates: $(2,2)$ and $(3,1)$, Module 8 will add $(3,0)$ and Module 9 will add $(3,-1)$ and $(2,-2)$. They will then send this vector to their respective parents 5, 4, and 7. Module 5 will add $(1,2)$, Module 7 will add $(1,-2)$, and Module 4 will not add anything since all its interfaces are connected. So on until this vector reaches the leader.

and the percentage of error detected (number of errors detected by our algorithm on the total number of errors in the system). Multiple simulations where done while varying different components at a time.

The first bash of simulations was for the same number of modules - In those simulations, the number of modules was 1000 -with different densities ranging from 0.18 to 1. All the shapes' configuration were randomly generated - For the exception of the cube since it is the only shape where the density is to its maximum-.

Figure 5 shows that when the density is increasing, the number of messages sent is also increasing. This is due to the fact that when the density is high, the modules are closer to each other, leading to an increased amount of neighbors per module wish will lead to a higher number of messages sent. Further more, Figure 5 shows that whenever the percentage of errors in a system is higher, the number of messages sent is lower. The reason behind this is that when modules are disconnected they can not send messages which will decrease the total number of messages sent.

The second bash of simulations were done for the same density while increasing the number of modules. The chosen density for this simulation was 1, for the reason that it was the value where the system was sending messages the most. Figure 6a shows how the number of messages is increasing linearly with the number of

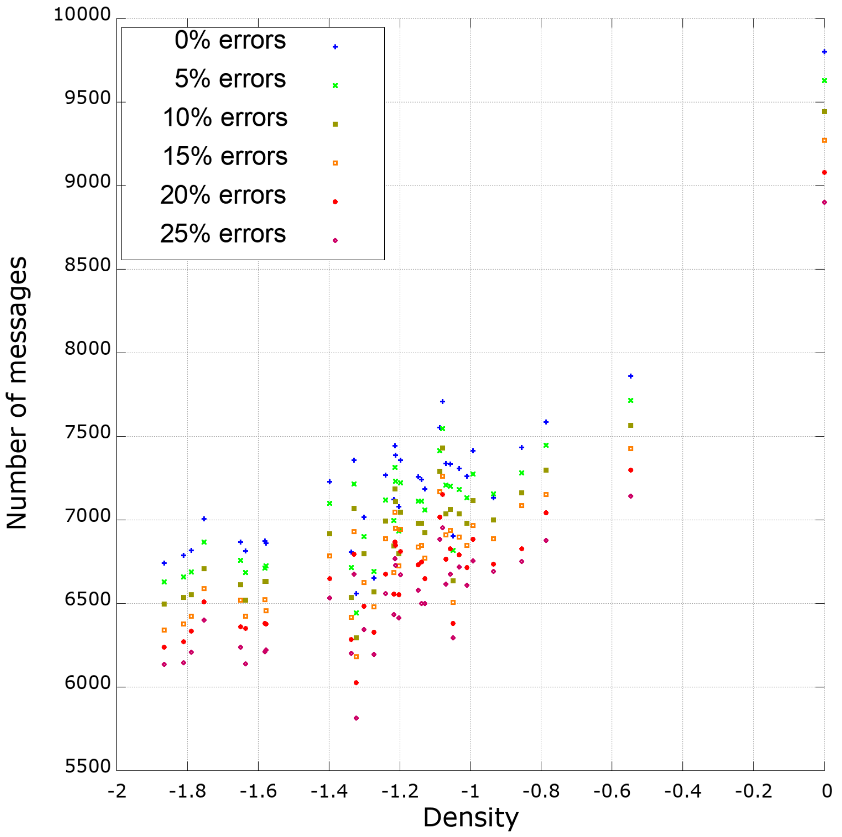
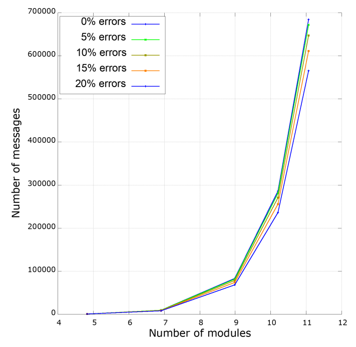


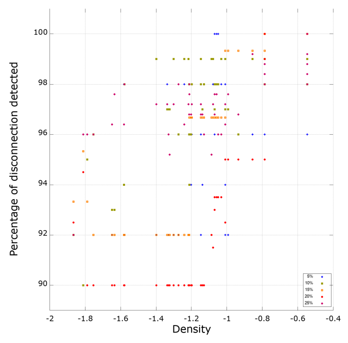
Fig. 5: This figure shows, on a logarithmic scale, how the number of messages sent in a 1000 3D system is changing with the density of the system. Each color represents the percentage of errors that existed in the system during the simulations. It is noticeable that the number of messages is increasing with the density. Furthermore, the number of messages decreases when the percentage of errors is increasing

modules. This shows that the complexity of the proposed algorithm, regarding the number of messages sent, is linear. It is also noticeable that when the number of errors is increasing in the system, the number of messages in decreasing. This is due to the fact that some messages are not able to be sent or delivered due to the errors encountered in the system.

During the previous simulations, the simulator collected the total number of errors detected by the proposed algorithm. Figure 6b shows the percentage of errors detected on the Y axes ranging from 90 to 100 against the density, on logarithmic scale, from -2 to -0.4. Five colors are shown on the graph. Each color represents the percentage of errors in the system. Figure 6b shows that for high densities, more errors are detected. A system with low density has less connected modules. Any occurring error might lead to a complete disconnection that is not detected by the proposed algorithm. Therefore, the higher the density the better the detection of errors is.



(a) This figure shows, on a logarithmic scale, the number of messages sent in the system according to the number of modules in a logarithmic scale.



(b) Percentage of errors detected.

Fig. 6: 2 Figures side by side

5 Conclusion

This paper presented Detector, a novel distributed algorithm for a hierarchical fault detection in modular robots. The proposed algorithm is able to detect partial disconnections in a modular robotic system. To achieve such task, the algorithm creates a spanning tree having the leader as root. Each module generates its coordinates starting from the leader as origin. Then, each module - starting from the leaves - will send the potential coordinates of none connected interfaces to its parent. The parent will also generates the potential neighbors' coordinates and add them to the received list, compare the list of potential neighbors' coordinates with the list of the actual coordinates. Any matching coordinates means a disconnection is found. Then, the parent, sends the remaining results to its parent, so on until it reaches the leader who will have the final decision regarding the existence of disconnections if any.

The simulation results show an elevated rate of disconnection detection specially with a high density system. The accuracy of finding disconnections was ranging from 90% to 100%. Furthermore, the simulations also showed that the complexity of the algorithm regarding the number of exchanged messages remains linear.

Future work should consider the ability to detect if any disconnection happened during the construction of the spanning tree, keep the spanning tree updated during the movement of the robots and also the ability to detect the complete disconnections in a modular robotic system. For example, the addition of different method of communications such a wireless communication can help detect a complete disconnection that leads to an isolation of some modules from the whole system.

Acknowledgements The authors would like to acknowledge the National Council for Scientific Research of Lebanon (CNRS-L) and the Holy Spirit University of Kaslik for granting a doctoral fellowship to Edy Hourany.

This work was partially supported by the ANR (ANR-16-CE33-0022-02), the French Investissements d'Avenir program, the ISITE-BFC project (ANR-15-IDEX-03) and the EIPHI Graduate School (contract "ANR-17-EURE-0002").

References

1. Saleh Ahmad, Hongwei Zhang, and Guangjun Liu. Distributed fault detection for modular and reconfigurable robots with joint torque sensing: A prediction error based approach. *Mechatronics*, 23(6):607–616, 2013.
2. Jad Bassil, Mohamad Moussa, Abdallah Makhoul, Benoît Piranda, and Julien Bourgeois. Linear distributed clustering algorithm for modular robots based programmable matter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pages 3320–3325. IEEE, 2020.
3. Andres Castano, Wei-Min Shen, and Peter Will. CONRO: Towards deployable robots with inter-robots metamorphic capabilities. *Autonomous Robots*, 8(3):309–324, 2000.
4. Jay Davey, Ngai Kwok, and Mark Yim. Emulating self-reconfigurable robots - design of the SMORES system. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4464–4469, Vilamoura-Algarve, Portugal, October 2012. IEEE.
5. Jean E Dunbar, Jerrold W Grossman, Johannes H Hattingh, Stephen T Hedetniemi, and Alice A McRae. On weakly connected domination in graphs. *Discrete Mathematics*, 167:261–269, 1997.
6. Edy Hourany, Christian Stephan, Abdallah Makhoul, Benoit Piranda, Bachir Habib, and Julien Bourgeois. Self-reconfiguration of modular robots using virtual forces. In *IEEE RSJ International Conference on Intelligent Robots and Systems (IROS 2021)*, Prague, Czech Republic, sep 2021.
7. M. W. Jorgensen, E. H. Ostergaard, and H. H. Lund. Modular ATRON: modules for a self-reconfigurable robot. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 2068–2073, Sendai, Japan, September 2004. ISSN: null.
8. Akiya Kamimura, Eiichi Yoshida, Satoshi Murata, Kohji Tomita, and Shigeru Kokaji. A Self-Reconfigurable Modular Robot (MTRAN) – Hardware and Motion Generation Software –. In *5th International Symposium on Distributed Autonomous Robotic Systems*, page 10, 2002.
9. Christopher G Langton. Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*, 22(1-3):120–149, 1986.
10. R. Moeckel, C. Jaquier, K. Drapel, E. Dittrich, A. Upegui, and A. Ijspeert. YaMoR and Bluemove — An Autonomous Modular Robot with Bluetooth Interface for Exploring Adaptive Locomotion. In M. O. Tokhi, G. S. Virk, and M. A. Hossain, editors, *Climbing and Walking Robots*, pages 685–692. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
11. Mohamad Moussa, Benoît Piranda, Abdallah Makhoul, and Julien Bourgeois. Cluster-based distributed self-reconfiguration algorithm for modular robots. In Leonard Barolli, Isaac Woungang, and Tomoya Enokido, editors, *Advanced Information Networking and Applications -*

Proceedings of the 35th International Conference on Advanced Information Networking and Applications (AINA-2021), Toronto, ON, Canada, 12-14 May, 2021, Volume 1, volume 225 of *Lecture Notes in Networks and Systems*, pages 332–344. Springer, 2021.

12. R Regin, S Suman Rajest, and Bhopendra Singh. Fault detection in wireless sensor network based on deep learning algorithms. *EAI Endorsed Transactions on Scalable Information Systems*, 2021.
13. John W. Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295, Tokyo, November 2013. IEEE.
14. D. Rus and M. Vona. A physical implementation of the self-reconfiguring crystalline robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 1726–1733, San Francisco, CA, USA, 2000. IEEE.
15. Behnam Salemi, Mark Moll, and Wei-min Shen. SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3636–3641, Beijing, China, October 2006. IEEE.
16. Pierre Thalamy, Benoit Piranda, Andre Naz, and Julien Bourgeois. Behavioral simulations of lattice modular robots with visiblesim. In *15th International Symposium on Distributed Autonomous Robotic Systems (DARS 2021)*, Kyoto, Japan, jun 2021.
17. Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Physica D: Nonlinear Phenomena*, 47(1):263–272, 1991.
18. Dong Xue, Qiang Lu, and Jian Li. Improving the morphology and control policy of self-reconfiguring modular robots in dynamic environment (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15939–15940, 2021.
19. M. Yim, D.G. Duff, and K.D. Roufas. PolyBot: a modular reconfigurable robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 514–20, 2000.
20. Youssef Mahmoud Youssef. Inducing rules about distributed robotic systems for fault detection & diagnosis. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1845–1847, 2021.