Self-Reconfigurable Robots Topodynamic

Samir Saidani
Laboratoire GREYC
Université de Caen
France
Email: saidani@info.unicaen.fr

***Abstract*— Modules connected to each other form a network. So, a modular robot is a module network. In the case of reconfigurable robots, the topology of this network evolves. We propose to ground the study of self-reconfigurable robots in a framework inspired by graph theory and cellular automata. We separate topological aspects from metrical ones, by defining the notion of graph topodynamic, and we provide a distributed algorithm which transforms a quadruped robot into a chain.**

I. Introduction

How to control the module network of a self-reconfigurable robot ? How to design a single unit so that a self-reconfigurable robot converges to the required shape ?

We are especially interested in distributed reconfiguration algorithm not requiring the exact description of the target shape. Thus Bojinov et al. (2000) [1], [2] proposed biologically inspired control algorithms for chain robots, using growth, seeds and scents concepts to make the target shape emerge from local rules. Another approach designed by Butler et al. (2003) [3] [4] is based on architecture-independent locomotion algorithms for lattice robots, inspired by the cellular automata model. Abrams and Ghrist (2003) [5] considered geometrical properties on a shape configuration space adapted to parallelization.

The difficulty to find a mathematical framework and general method able to address the shape controlling problem is due in our opinion to the metrical, topological and distributed nature of a self-reconfigurable robot.

For instance, two modules can be near from each other, but not connected, and models based on a 2D or 3D array representation are not always accurate. How to distinguish neighbor connected modules from neighbor disconnected modules ? We propose in the first stage of the modular robots modelling to separate the topological aspect from the metrical one, and to study how we can express the topological nature of a self-reconfigurable robot, considered as more fundamental than its metrical aspect. Indeed, if a reconfiguration is impossible in a topological manner, then it will be impossible in a metrical way ; the opposite is false.

Modular robots are modules networks , and networks are usually modeled by graphs, ideal to stress the relation between entities. We can for example express the modules connectivity by bounding the degrees of a graph, unidirectional and bidirectional connections by directed or undirected edges... However, the reconfiguration of a modular robot implies the evolution of the modules network topology, and thus of its underlying graph.

Modelling the evolution of a network topology is quite hard to capture in graph theoretic model. The fundamental work on random graphs, by Erdös and Rény [6] was the very first attempt to add dynamic in a graph. Recently, interest has grown among graph theoricists in dynamic graphs, and especially in dynamic algorithms able to incrementally update a solution on a graph while the graph changes. To represent a dynamic graph, Harary (1997) [7] proposed dynamic graph models based on logic programming and the study of the sequence of static graphs. Ferreira (2002) [8] recently proposed a model called evolving graph, whose definition is based upon an ordered sequence of subgraphs of a given digraph. But this given digraph induces an a priori knowledge on the dynamic process.

Moreover, we would like the graph topology to evolve in a decentralized way with local and simple rules : each module has a local knowledge on its environment and a limited power for computation. Cellular automata are well known for their ability to express complex dynamics from the local knowledge of the cells. The underlying lattice of a cellular automata is usually static, but Ilachinski and Halpern (1997) [9] developped a cellular automata model in which the underlying d-dimensional array evolves according to link transition rules. Unfortunately, this model is expressed in the metric space $\mathbb{Z}^d$ and link transition rules depend on the states of cells neighborhood.

To combine graph theory expressivity with richness of cellular automata dynamic, the first section defines the notion of graph topodynamic, with the assumption that a module only knows about its neighborhood and the neighbors of its neighborhood.

The second part is devoted to the construction of a topodynamic which converges to a target topology by emergent calculus : the modules do not know the goal configuration and the final configuration emerges from the modules collective behavior.

By this way, we hope to transform the shape controlling problem into the study of graph topodynamic, namely the discovering of topodynamics converging towards a given topology.

II. Graph Topodynamic

We first remind basic notions in graph theory and then state a topological definition of a graph independent of its embedment in metric space. We finally define the notion of graph topodynamic.

A. Preliminaries

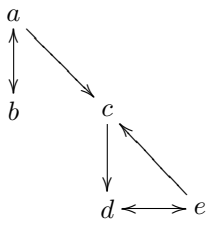
Definition 2.1 (Graph): A graph is a pair (V, E) with V a finite set of vertices and E a set of edges, finite subset of $V \times V$.

A graph is undirected if the relation defined on V is symmetric, otherwise the graph is directed, and edges have a direction. The *order* of a graph is its number of vertices $|V|$. Two vertices are said to be *adjacent* if they are joined by an edge.

The *neighborhood* of a vertex v is the set $\tau(v)$ of vertices x such that x is adjacent to v. If there is no ambiguity with the context, we note a neighborhood $\tau(v) = \{x, y, \dots, z\}$ by $xy \dots z$. The *out-neighborhood* of a vertex v is the set $\tau^+(v)$ of vertices x outgoing from v. The *in-neighborhood* of a vertex v is the set $\tau^-(v)$ of vertices x pointing to v. We see that an undirected graph (resp. directed graph) is completely describe by giving the set of its vertices and a neighborhood (resp. in-neighborhood *or* out-neighborhood) on each vertex. The *degree* (resp. *indegree*, *outdegree*) of a vertex is the number of its neighbors (resp. in-neighbors, out-neighbors). The degree of a graph, noted $d^\circ(G)$ is the maximum degree of a vertex. Note that for an undirected graph, each edge v, w could be considered as a double arrow (v, w) and (w, v), so the in-neighborhood is equal to the out-neighborhood of a vertex.

Definition 2.2 (Graph Topology): The topology τ (resp. τ^+, τ^-) of a graph G is the family of its neighborhoods $(\tau_v)_{v\in V}$ (resp. out-neighborhoods $(\tau_v^+)_{v\in V}$, in-neighborhoods $(\tau_v^-)_{v\in V}$).

Example 2.3: Let G be the following graph :



The neighborhood $\tau(a)$ of a is $\{b, c\}$ or in short bc. We have too : $\tau^+(a) = bc$,$\tau^-(a) = b$,$\tau(c) = ade$, $\tau^+(c) = d$,$\tau^-(c) = ae$, $d^\circ(G) = 3$ because $|\tau(c)| = 3$

B. Topodynamic

Let us now define the notion of a sequence of graphs.

Definition 2.4 (Sequence of Graphs): A sequence of graphs is a family of graph $(\mathcal{G}_i)_{i\in\mathbb{N}}$ with $\mathcal{G}_i = (V_i, \tau_i)$.

For simplicity, we will consider from now only sequence of graphs with constant order, i.e $\forall i \in \mathbb{N}$, $V_i = V_0$.

What is the difference between a sequence of graphs and a dynamic graph ? Usually, a dynamic system is characterized by its transition function : we can compute the state of the system from an initial state and past states. Basically, a sequence of graphs can not change its topology on its own : the evolution of the topology is predetermined by giving a family τ_i of topologies. However, we can associate a transformation function to a sequence of graphs. So we will call dynamic graph a sequence of graphs consisting of an initial graph and a function which transforms its topology to a new topology.

Definition 2.5 (dynamic graph - global transition function): A dynamic graph is the pair (G_0, Δ), such that $G_0 = (V, \tau_0)$ is an initial graph and $\Delta : (V \mapsto 2^V) \mapsto (V \mapsto 2^V)$ define a *topodynamic* by mapping a topology on V to a new topology.

Nevertheless, we would like to have dynamic graph vertices more active than in a sequence of graphs, namely able to change their own degrees by accepting, keeping or removing its adjacent edges, according to local transition rules inducing the graph topodynamic. Local transition rules is widely used in cellular automata area : the state of an automaton depends on its own state and the state of its neighbors. Local transition function, simultaneously applied to each cell, determine the dynamic of a cellular automata. Although cellular automata are usually defined on regular lattices, this definition can be extended to more complicated graph : graph of automata (connected bounded degree graph), first introduced by Rosenstiehl (1966) [10]. In a graph of automata, each node have a state and the next state depends of its current state and the state of its neighbors.

Definition 2.6 (graph of automata): A graph of automata is a triplet (S, G, δ) where S is a finite set called set of states, $G = (V, \tau)$ is a graph, $\delta : S \times \{(S \cup \epsilon)^{d^\circ(G)} / \sigma\} \mapsto S$ is the transition function where ϵ is a special element used when the vertex has less than the maximum degree of the graph. σ is the equivalence relation defined on the cartesian product S^n with $x\sigma y$ if x is a permutation of y. So S^n / σ is the unordered set S^n.

In this definition of graph automata , the underlying graph is static. We study here the possibility to have an evolving underlying graph : this evolution may be controlled by active vertices, kind of automata able to connect and disconnect their own edges in the network. We give to the automata the control of its underlying graph by slightly modifying the graph automata definition as following.

Definition 2.7 (dynamic graph - local transition function): A dynamic graph is the pair (G_0, δ) where $G = (V, \tau_0)$ define an initial graph, and $\delta : S \times \{(S \cup \epsilon)^{|V|-1} / \sigma\} \mapsto S$ with $S = 2^V$ the set of states, define the local transition function, where ϵ is a special element used when the vertex has less than $|V| - 1$ (the maximal degree of the dynamic graph).

A node chooses its next neighborhood according to its current neighborhood and the current neighborhood of its neighbors. If we replace "neighborhood" in the precedent sentence by the word "state", we retrieve the usual definition of the evolution of a cell in cellular automata.

In order to deal with the different neighbors of a given neighborhood, we build from the application τ an application $\vec{\tau}$ which for each vertex gives its *neighbors vector* : $\vec{\tau} : \{v\} \mapsto (\{1,\ldots,|\tau(v)|\} \mapsto V)$ such that $\bigcup_{i=1}^{|\tau(v)|} \vec{\tau}(v)(i) = \tau(v)$ where $v \in V$

If a vertex v has a degree n, its next state $\tau_{i+1}(v)$, namely its next neighborhood, is given by :

$$\tau_{i+1}(v) = \delta(\tau_i(v), \tau_i(\vec{\tau_i}(v)(1)), \ldots, \tau_i(\vec{\tau}(v)(n)), \epsilon, \ldots, \epsilon)$$

We define now the fixpoint topology for a given topodynamic as a graph topology unchanged by applying this topodynamic.

Definition 2.8 (fixpoint topology): A fixpoint topology τ for the topodynamic Δ is a topology such that $\Delta(\tau) = \tau$

A question we may ask is for which topodynamic a given topology is the fixpoint.

III. Fixpoint Topology

In this section, we show how to apply the graph topodynamic model through the example of a tree-to-chain reconfiguration. We provide a distributed algorithm defining a topodynamic on an initial tree, converging to a chain, the fixpoint topology of this dynamic.

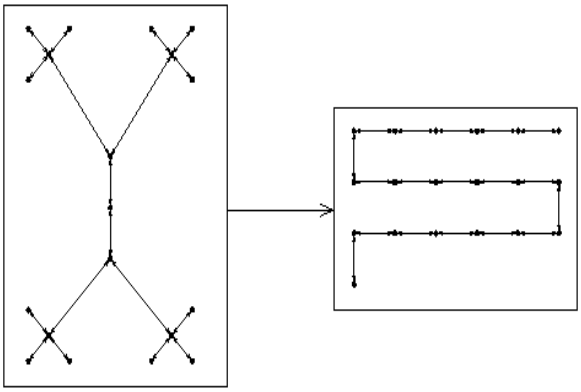


Fig. 1. Tree to chain reconfiguration

Let us imagine that we want reconfigure a quadruped robot (as the conro [11] or polybot robot [12]), represented to the left of the figure 1, to a caterpillar robot. Each node represents a module, and directed edges the connection between modules.

There are several principles behind a topodynamic design :

- Reconfiguration : the reconfiguration is done thanks to the knowledge of a node and of the neighborhood of its neighbors. It can disconnect from its current nodes to reconnect itself to a neighbor of its neighbors.
- Local knowledge : A node knows its own indegree and outdegree (computed from its knowledge of its in and out-neighbors) and the in and outdegrees of its direct neighbors (computed from its knowledge of in and out-neighbors of its neighbors).
- Outgoing connection control : A node only controls its outgoing connections, it cannot decide to disconnect itself from an ingoing connection but can connect to or disconnect from its outgoing connections.
- Decision process : To take a decision, a node may exploit the dissymmetry of its neighborhood. For that reason, we avoid the cycles in a graph because of the possibility to lose graph connexity.
- Connexity : A node must never be isolated during the reconfiguration process.
- Uniformity : All nodes have the same set of rules.
- Synchronicity : The computation and reconfiguration are totally synchronous among the nodes.

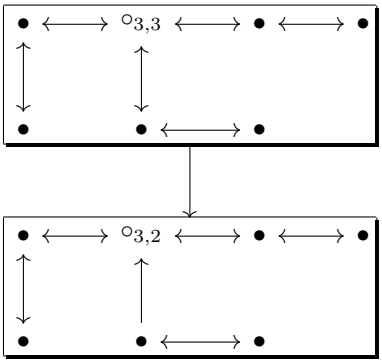
Let us now consider the dynamic graph $\mathcal{D}$ defined as following :

- G_0 is the tree as defined in the figure 1,
- the topodynamic is defined by the distributed algorithm below.

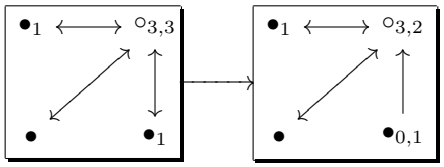
```
Tree-to-Chain(v)
 1  switch
 2    case d+(v) ≥ 3 :
 3         if disconnectNodeOfOutDegree(1)
 4           then
 5                disconnectNodeOfOutDegree(2)
 6         return
 7    case d−(v) = 0 & d+(v) = 1 & |τ+(τ+(v)| ≥ 2 :
 8         if d(τ+(v)) ≥ 2
 9           then
10                connectToOutNgbOfMyNgb
11         return
12    case 0 ≤ d−(v) ≤ 1 & |τ(v)| = 2 :
13         w ← τ+(v) ⊖ τ−(v)
14         for u ∈ w
15           do
16              if d+(u) ≥ 2
17                then
18                     connectToOutNgbOfNgb(u)
19              return
20    case 1 ≤ d−(v) ≤ 2 & d+(v) = 1 & |τ(v)| = 2 :
21         connectToNodeOfInDegree(0)
22         return
23    case d(v) ≥ d+(v) + 2 :
24         connectToOneOfMyStrictInNgb
25         return
```

The function `disconnectNodeOfOutDegree(`n`)` tries to disconnect the current node from a neighbor of outdegree n and return true. If it fails, return false. The function `connectToNgbOfMyNgb` connect the current node to somekindof neighbor of its neighbor. The function `connectToNgbOfNgb(`v`)` connect the current node to somekindof neighbor of its neighbor v. The function `connectToNodeOfInDegree(`n`)` connect the current node to a node of indegree n. The function `connectToOneOfMyStrictInNgb` connect the current node to one of its strict in-neighbors, namely in-neighbors which are not out-neighbors.

We can now compute the graph topodynamic. The cases below are encountered during the reconfiguration process. To trace the computation, we distinguish the current node (in white) from the other nodes (in black). For each node, the first indice gives the indegree of the current node, and the second indice the outdegree. If there is only one indice, this is the degree of the node. We call a connection closed if the edge is bidirectional, and we call second neighbors the neighbors of a node neighbors.



(a) Disconnect node of outdegree 2



(b) Disconnect node of outdegree 1

Fig. 2. Outgoing connection disconnection

Lines 2-6, figure 2 : if a node is connected to too much nodes ($d^+(v) \geq 3$), then it tries to disconnect first to node of outdegree 1, then to disconnect to node of outdegree 2.

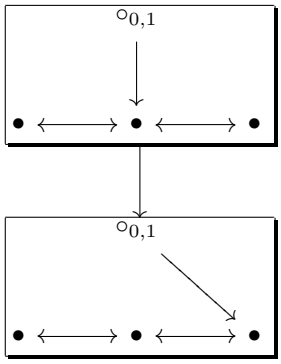
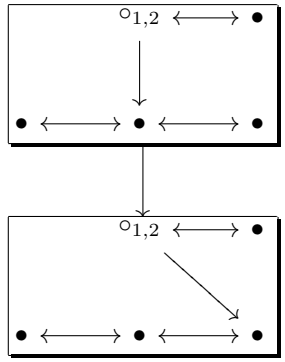
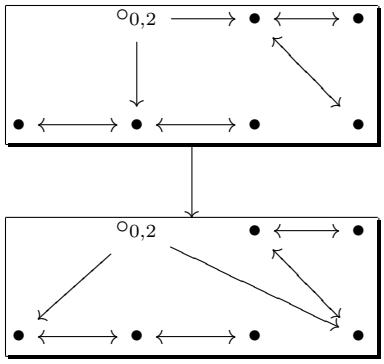


Fig. 3. Reconfiguration

Lines 7-11, figure 3 : If the indegree of a node is zero and the outdegree is one, then this node is necessarily the extremity of a chain. If the outdegree of its unique neighbor is two, then the node tries to connect to a random second out-neighbors. In fact, the node tries to find the extremity of a chain.



(a) One closed connection for the current node



(b) No closed connection for the current node

Fig. 4. Another reconfiguration

Lines 12-19, figure 4 : This node is necessarily between two nodes, it will try to find the extremity of a chain by exploring its second neighbors.

Lines 18-20, figure 5 : if a node is inside a chain and its indegree is one, then it will close the connection with its in-neighbor.

Lines 23-25, figure 6 : A node will try to have only one unclosed connection. We verify that no rules apply on a chain graph, therefore the chain topology is a fixpoint topology for the algorithm TREE-TO-CHAIN.

IV. CONCLUSION

This work is a proposal of a framework based on graph topodynamic and cellular automata intended to address the problem of controlling modules network topodynamic.

In the first part, we defined the topology of a graph and proposed to make a distinction between sequence of graphs and dynamic graphs: a dynamic graph is a sequence of graphs where a local or global topodynamic determines the topological evolution of a graph.

In the second section, we showed through an example that a topodynamic can be defined from the local knowledge of

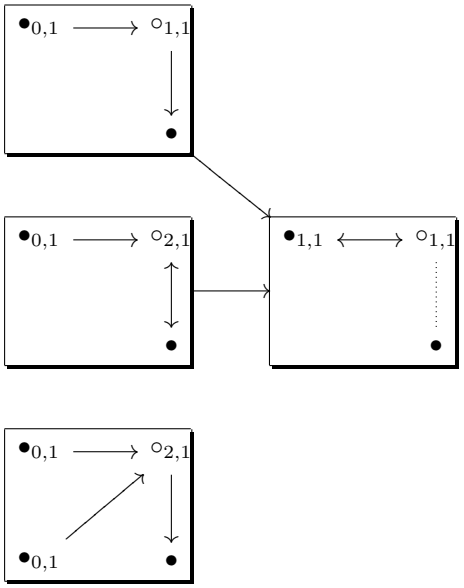


Fig. 5. Closure

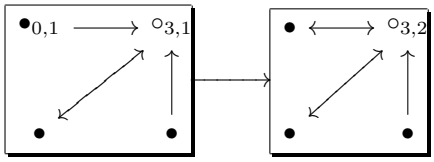


Fig. 6. Only one unclosed

each vertex : we built a dynamic graph able to change its own underlying graph according to local and simple rules.

Clearly more work needs to be done to extend this model. First of all, a node does not have the memory of nodes already explored during random reconnections, which leads to a low convergence speed. Moreover, this framework is currently unable to deal with cycles in graph : cycles are totally symmetrical configurations and our approach is based on the exploitation of asymmetrical local configurations. Breaking symmetries of a cycle demands to enlarge the model by considering dynamic graphs of automata, i.e dynamic graphs in which each node has a state. Another way to extend this model is to allow a node to know the neighborhood of the neighborhood of its neighbors, hence to increase the convergence speed. Notice that a reconfiguration process involving compression, expansion and any deformation of modules is not captured by this model : we have to add a metric distance on the graph topology to model such transformations.

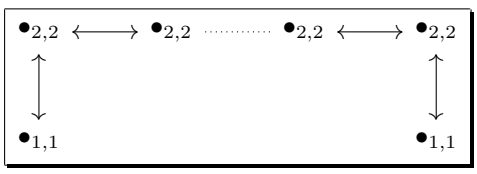


Fig. 7. Chain topology

We hope that this work is a first stage and a basis for a more general framework sufficiently powerful to express any kind of reconfiguration process. We are currently working on the proof that, given any acyclic graph, the topodynamic described in this paper converges to a chain shape. Other useful topodynamics remain to study : from a connexe graph to a chain graph, from a lattice to a chain , from a chain to a lattice, and so on. A simulation implemented in Smalltalk is underway with a view to help us in rules discoveries involved in the emergence of different network topologies.

ACKNOWLEDGMENT

This work was supported by an MENRT Research Studentship, and is a part of the MAAM Project [13]. We thank François Bourdon and Serge Stinckwich for their comments and to allow us to participate in the MAAM project. We also thank Abdel-illah Mouaddib and Elias O'Regan for their valuable advices.

REFERENCES

[1] H. Bojinov, A. Casal, and T. Hogg, "Multiagent control of self-reconfigurable robots," June 20 2000, comment: 15 pages, 10 color figures, including low-resolution photos of prototype hardware. [Online]. Available: http://arXiv.org/abs/cs/0006030

[2] ——, "Emergent structures in modular self-reconfigurable robots," in *Proceedings, IEEE Int. Conf. on Robotics & Automation (ICRA'00)*, vol. 2, San Francisco, California, USA, 2000, pp. 1734 –1741.

[3] Z. Butler and D. Rus, "Distributed planning and control for modular robots with unit-compressible modules," *International Journal of Robotics Research*, vol. 22, no. 9, pp. 699–716, September 2003.

[4] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic decentralized control for a class of self-reconfigurable robots," in *Proceedings, IEEE Int. Conf. on Robotics and Automation (ICRA'02)*, Washington, DC, USA, 2002, pp. 809–815.

[5] A. Abrams and R. Ghrist, "State complexes for metamorphic robots," *International Journal of Robotics Research*, 2003, in press.

[6] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, pp. 17–61, 1960, a seminal paper on random graphs. Reprinted in *Paul Erdős: The Art of Counting. Selected Writings*, J.H. Spencer, Ed., Vol. 5 of the series *Mathematicians of Our Time*, MIT Press, 1973, pp. 574–617.

[7] F. Harary and G. Gupta, "Dynamic graph models," *Math. Comput. Modelling*, vol. 25, no. 7, pp. 79–87, 1997.

[8] A. Ferreira, "On models and algorithms for dynamic communication networks: The case for evolving graphs," in 4^e *rencontres francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL'2002)*, Mèze, France, May 2002.

[9] Ilachinski and Halpern, "Structurally dynamic cellular automata," *COMPSYSTS: Complex Systems*, vol. 1, 1987.

[10] P. Rosenstiehl, "Existence d'automates finis capables de s'accorder bien qu'arbitrairement connectés et nombreux," *International Computer Science Bulletin*, vol. 5, pp. 245–261, 1966.

[11] K. Støy, W.-M. Shen, and P. Will, "How to make a self-reconfigurable robot run," in *Proceedings, First Int. Joint Conf. on Autonomous Agents & Multiagent Systems (AAMAS'02)*, Bologna, Italy, 2002, pp. 813–820.

[12] M. Yim, D. Duff, and K. Roufas, "Polybot: a modular reconfigurable robot," in *Proceedings, IEEE Int. Conf. on Robotics & Automation (ICRA'00)*, vol. 2, San Francisco, California, USA, 2000, pp. 1734 – 1741.

[13] D. Duhaut, "Maam project," 2001, *http://www.laas.fr/robea/maam.html*.