Evolution of Shape-Changing and Self-Repairing Control for the ATRON Self-Reconfigurable Robot

Christensen, David Johan

Published in:
Proceedings of the 2006 IEEE International Conference on Robotics and Automation

Link to article, DOI:
10.1109/ROBOT.2006.1642084

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Evolution of Shape-Changing and Self-Repairing Control for the ATRON Self-Reconfigurable Robot

David Johan Christensen
Maersk Mc-Kinney Moller Institute for Production Technology
University of Southern Denmark, Odense, Denmark
Email: david@mip.sdu.dk

***Abstract*— The ATRON self-reconfigurable robot consists of simple interconnected modules. Modules move relative to other modules and as a result change the shape of the robot. The ATRON modules are difficult to control because of complex motion constraints on the modules. Motion constraints are reduced by using meta-modules composed of three modules. A meta-module may emerge from unstructured groups of modules if three modules are connected in the right configuration. The meta-module then moves on a surface of modules and stop at another position. To attract moving meta-modules and thereby to specify the shape-changing task of the robot we use attraction-points. In this work we evolve a distributed artificial neural network controller for the modules. The controller is identical on every module and controls when a meta-module emerges, how it move and when it stops. In simulation we demonstrate how this control strategy allows the ATRON robot to shape-change to support an unstable roof, build a bridge across a gap and to self-repair a broken bone. We conclude that the control strategy is able to shape-change and self-repair the ATRON robot independent on whether it consists of dozens, hundreds or thousands of modules.**

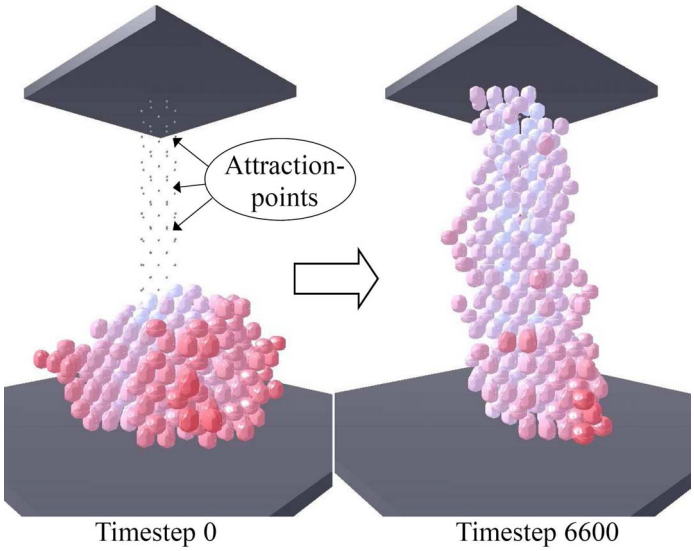


Fig. 1. In order to support an unstable roof the structure of 500 ATRON modules shape-change, stretching upwards to achieve the functionality of a pillar. The process is guided by attractions-points which are shown as small spheres.

I. Introduction

Self-reconfigurable robots consist of a number of interconnected heterogeneous or homogeneous robot modules. Modules have their own processing power and are able to communicate with other modules and sense the environment. In lattice-based self-reconfigurable robots the modules are rigidly interconnected in a lattice structure. The modules are able to connect to, disconnect from and move relative to other modules in order to change the configuration of modules. The modules are designed to give the self-reconfigurable robot abilities such as shape-change, self-repair and self-assemble. Such abilities may produce exceptional robots in terms of flexibility, versatility and robustness.

In this work we consider the ATRON module [8] which is latticed-based and able to self-reconfigure in 3D. An ATRON module has a limited mobility in a structure of modules. Because of complex motion constraints on the modules, moving a module from one lattice position to another is difficult or may even be impossible. Therefore the self-reconfiguration of ATRON modules is non-trivial.

We consider shape-changing structures of more than 50 ATRON modules. The desired functionality of the robot system is designed by placing virtual points in space, called attraction-points. Our goal is to build a controller, that is distributed, identical on every module and makes the robot change its shape towards the attraction-points.

To overcome modules' limited mobility, we use three modules that cooperate as a single entity, a meta-module, to move across the surface of other modules. The mobility of such a meta-module is much higher that the mobility of an individual module.

Attraction-point triggers unstructured groups of passive modules to form meta-modules which then *emerge*. Meta-modules *move*, towards attraction-points, across the surface of passive modules. Meta-modules which get stuck or reach an attraction-point *stop* moving and the modules comprising the meta-module may at a later time be part of another meta-module. To move meta-modules perform meta-actions which is a sequence of basic module actions.

Knowing its local surrounding a meta-module may calculate a subset of its reachable space, which is a graph defining the possibilities of the meta-module in its current situation. Using this graph a meta-module can calculate the shortest path of meta-actions from its current state (position and orientation) to another state in its local

0-7803-9505-0/06/$20.00 ©2006 IEEE

surrounding. Based on this modules decide when to emerge a meta-module, how it should move and when to stops. The part of the modules' controller that makes these decisions are three small artificial neural networks (ANN). The ANN's takes input calculated from the subset of the reachable space of the meta-module. As outputs one ANN gives a decision on whether to emerge. A second ANN gives as output whether to stop. The third ANN assigns a fitness value to every state in the known subset of the reachable space graph. The meta-module will move towards the fittest state. We evolve the weights of the ANN controller by letting the system shape-change and measure its performance as a fitness value.

The combination of ATRON modules, meta-modules, attraction-points and evolved artificial neural network control gives rise to a robot which can change its shape and self-repair in a large range of scenarios. This slightly complex control strategy is the best known solution to the complex problem of shape-changing and self-repairing large structures of ATRON modules.

II. Related Work

Besides the ATRON module, hardware prototypes of modules able to self-reconfigure in 3D include the MTRAN [14], 3D-Unit [13], Molecule [10] and I-Cubes [22]. One main difference between ATRON and these systems is the complexity of the individual module in term of degree of freedom. The ATRON module's single degree of freedom may make it simple to manufacture but hard to control.

Most prior work, on shape-change of self-reconfigurable robots, focus is on achieving a particular target shape. This is problematic to achieve for systems such as the MTRAN and ATRON, because of difficult motion constraints on the modules. Inspired by Bojinov et al. [3] we avoid this problem by trying to achieve a particular functionality instead of a particular target shape.

Controlling shape-change of large groups of modules makes direct search strategies infeasible because of the computational complexity involved. Planning strategies are often possible on smaller groups of modules, to solve a sub-problem or using heuristic search [10], [21], [24]. Distributed control [4], [5], [11], [16] strategies are independent of the global properties of the structure of modules. This helps to ensure robustness, but distributed control may be harder to design than centralized control.

In [1], [6], [12], [17], [18], [21], [23] groups of modules are used as meta-modules to handle the base modules' motion constraints. A negative characteristic of meta-modules is that they increase the granularity of the system. Also the increase in cost and complexity of a single meta-module, compared to a single module, might be difficult to justify with the improved mobility.

Prior work on self-repair in self-reconfigurable robots generally involves the detection of module failure, decisions on how to remove a defect module and how to replace it with a spare module [7], [20]. Alternatively, as in this work, self-repair can emerge as a side effect of the control instead of having a specialized self-repairing part of the controller [19].

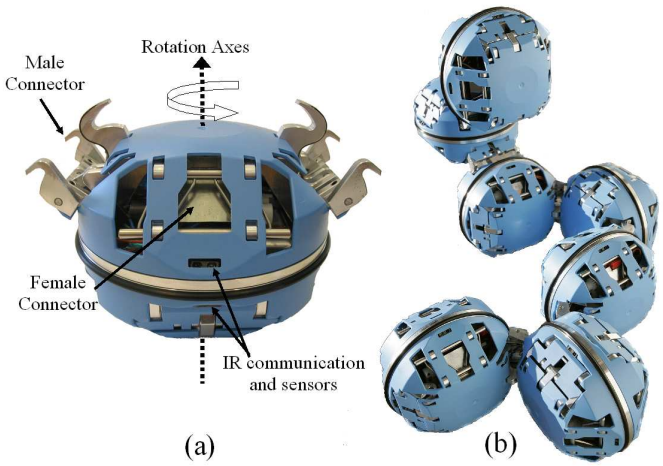


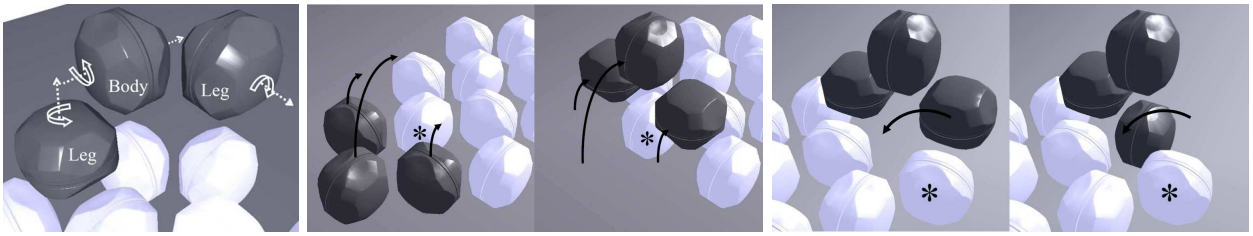
Fig. 2. Photographs of: (a) A single ATRON module, on the top hemisphere the two male connectors are extended on the bottom hemisphere they are contracted. (b) A structure of seven ATRON modules connected in the surface-centered cubic lattice structure.

Artificial evolution has previously been used on self-reconfigurable robots to automate the design of control. Østergaard et al. [15] evolved with limited success distributed state-machine based controllers for small structures of 12 to 20 ATRON modules. Similarly evolution was used on small groups of MTRAN modules to automatically generate locomotion patterns [9]. For the 2D metamorphic system genetic programming was used to generate controllers for movement of 8 modules to solve tasks such as moving through a narrow passage [2].

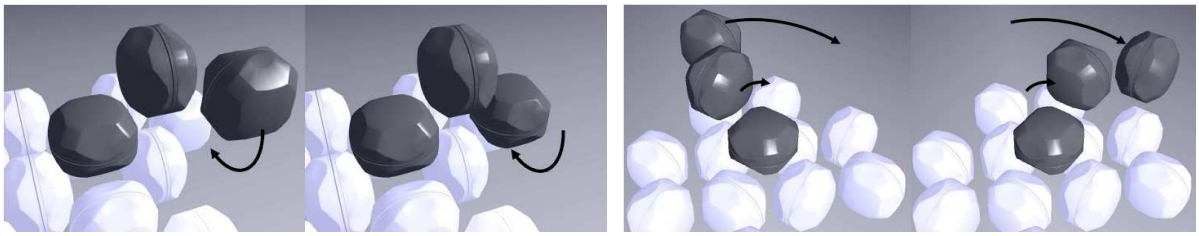
III. The ATRON Self-Reconfigurable Robot

The ATRON module, has a single rotational degree of freedom and is able to self-reconfigure in 3D, see figure 2(a). It has a spherical appearance composed of two hemispheres, which the module can actively rotate relative to each other. The modules connect to neighbour modules using its four actuated male and four passive female connectors. The connectors are positioned in 90 degrees intervals on each hemisphere. Using infrared channels the module is able to communicate with neighbour modules and sense distance to nearby obstacles or modules. Two "Atmel ATmega128" microcontrollers, one on each hemisphere, controls the module. A module weighs 0.850kg and has a diameter of 110mm. 100 hardware prototypes of the ATRON modules exist. A more extensive description of the ATRON hardware can be found in [8]. In this work the modules are always connected in a surface-centred cubic lattice structure, see figure 2(b).

Motion constraints on the modules affect their ability to self-reconfigure. The single rotational degree of freedom of a module makes its ability to move very limited, in fact the module morphology does not allow it to move by itself. One module may move another module by rotating it while



(a) ATRON Meta-module. (b) Meta-action: Turn around corners. (c) Meta-action: Shifting orientation.



(d) Meta-action: Rotation of body-module. (e) Meta-action: Rotation of leg-module.

Fig. 3. Illustrations of the morphology of the ATRON meta-module and its meta-actions. The dark modules comprise the meta-module. The $*$-marked modules in (b) and (c) are required to participate in the corresponding meta-actions.

they are interconnected. Before a module disconnects a neighbour module it must make sure that no modules will fall off the structure. When rotating a module must take into account its limited actuator strength and avoid collisions. A single module has the strength to rotate one or two other modules in any direction (worst case is against gravity).

IV. The ATRON Meta-Module

Motion constraints of the ATRON modules may to some extend be reduced by the use of meta-modules [6]. In this work we consider meta-modules composed of three ATRON modules: one centre module (body) is connected to two other modules (legs), one on each hemisphere, see figure 3(a). To move meta-modules perform meta-actions, which consist of a sequence of connections, disconnections and ± 90 degree rotations. The meta-module is able to perform twelve different meta-actions. The meta-actions follow three different blueprints which are explained below:

Blueprint 1 (8 meta-actions): Meta-actions following this blueprint allow the meta-module to move as a two legged walker on a flat surface of modules. First, the meta-module connects to a structure-module using a hemisphere of a leg which is not connected to the body. Second, the meta-module is disconnected from all other modules. Third, either the connected leg-module or the body-module makes a ± 90 degree rotation, as illustrated in figure 3(e) and 3(d).

Blueprint 2 (2 meta-actions): The $*$-marked module in figure 3(b) is required to help the meta-module when performing this meta-action. The four modules perform a sequence of connections, disconnections and rotations, which makes it turn around a "corner" as illustrated in figure 3(b). A different meta-action following the same blueprint allows it to turn around a corner in the opposite direction.

Blueprint 3 (2 meta-actions): The $*$-marked module in figure 3(c) is required to rotate one leg-module towards the other leg-module, which then becomes the new body-module of the meta-module. Similarly a meta-action that rotates the other leg follows this blueprint. The effect of these meta-actions are to shift the orientation of the meta-module as illustrated in figure 3(c).

The combination of morphology and meta-actions provides the meta-module with a high ability to move on the surface of other modules.

V. Attraction-points as Task Specification

We use attraction-points to control the flow of meta-modules from one place to another on the structure of modules. Attraction-points are virtual points in space, whose positions are known to the modules. Meta-modules are attracted by attraction-points and move toward them if possible. Attraction-points are used to specify tasks for the self-reconfigurable robot. E.g. if the robot is to change its shape to meet some specifications this could be done by providing the robot with a set of attraction-points in the desired shape. Two types of attraction-points have been used in this work: inhibiting and non-inhibiting. An inhibiting attraction-point *turns off* if a module is placed at its position, then meta-modules are no longer attracted by that point. Non-inhibiting attraction-points always attract meta-modules.

VI. Artificial Neural Network Controller

Modules have to make decisions concerning:

1) When should the module emerge as part of a meta-module?
2) Which meta-actions should the meta-module perform?
3) When should the meta-module stop?

Decisions are made by three feedforward, 3-layer, sigmoid activation function, artificial neural networks(ANN),

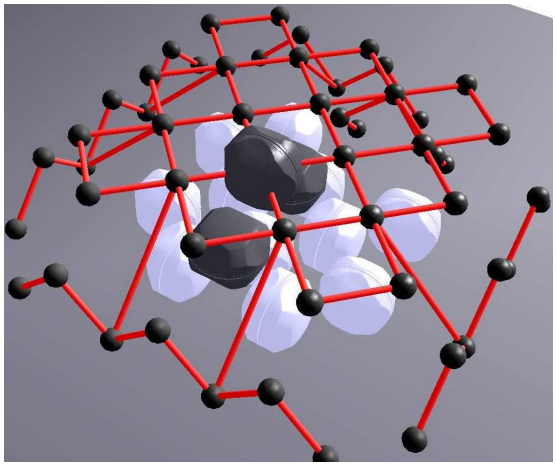


Fig. 4. The illustration shows the reachable-space graph of a meta-module on a structure of modules. The reachable-space defines for any reacheable state of the meta-module which meta-actions it may perform and the corresponding effect on its state. Small spheres are vertices and lines are edges in the graph. For simplicity in this illustration only the positions of the centre body-module are used to visualize the states.

one for answering each of the questions. The controller calculates at runtime a number of inputs to the ANN's. The inputs are calculated based on positions of attraction-points and the state of the local surrounding, such as positions and orientation of nearby modules and obstacles. In subsection VI-A, we explain how the meta-modules calculate a subset of their reachable-space from which the inputs to the ANN's are calculated. In subsection VI-B, VI-C and VI-D we explain how the ANN's are used to control the meta-modules.

A. Reachable-space of Meta-module

Most of the inputs given to the ANN's are related to the reachable-space of a meta-module:

- The *reachable-space* of a meta-module is a graph, where vertices are legal states(position and orientation) of the meta-module and edges are legal meta-actions which brings the meta-module from one legal state to another.

Since both the meta-module and the environment are changing dynamically so will its reachable-space. An example of a reachable-space of a meta-module on a structure of modules is illustrated in figure 4. Meta-modules calculate at runtime a small subset of their reachable-space, to produce inputs for the ANN's: 1) The meta-module builds a map of the local surroundings using neighbour to neighbour communication. The communication range is limited to six neighbours away. The map contain informations about which positions, in the ATRON lattice, are known to contain passive modules, modules part of a meta-module, obstacles and which positions are empty. 2) The reachable-space subset is calculated in a breadth-first manner. Initially the subset only contains one state - the actual state of the meta-module. Iteratively the reachable-space subset is expanded by repeatedly applying rules to the states in it. Rules correspond to meta-actions, so in total there are twelve rules one for each meta-action. A rule has a pre-condition which states which positions relative to the meta-module should be empty, which should contain passive modules and constraints on the orientation of the meta-module (for some meta-actions). A rule also has a post-condition which give the new state of the meta-module relative to the old one. To avoid computational explosion, states already seen are not recalculated and a fixed number (12) of iterations on the graph are done. This keeps the size of the graph down. Based on 5000 test samples there are on average 83 and a maximum of 687 vertices in the graph. The relative small size of the graph enables it to be calculate at runtime on the ATRON modules.

B. Emergence of Meta-module

With a low probability a passive module will attempt to emerge a meta-module. It randomly selects two passive neighbour modules, one on each hemisphere. It then calculates the reachable-space subset of that meta-module. From this it calculates the following inputs to an ANN which have 3 neurons in input-layer, 3 neurons in hidden-layer and 1 neuron in output-layer:

- Distance to nearest attraction-point from current state of the meta-module.
- Distance to nearest other meta-module from current state of the meta-module.
- Biggest known possible reduction in the distance between the meta-module and its nearest attraction-point.

The distance is calculated as the sum of the Euclidian distances for the three modules in the meta-module. If the output value of the ANN is greater than 0.5 the meta-module will emerge and the ANN's for movement and stopping will control the meta-module.

C. Movement of Meta-module

When a meta-module has emerged it starts to move, by selecting one of the twelve meta-actions. The selected meta-action is then performed and the process is repeated. An ANN with 4 neurons in input-layer, 4 neurons in hidden-layer and 1 neuron in output-layer is used to select which meta-action to perform next. Each state in the reachable-space subset is evaluated separately. We make extensive use of the shortest path sequence of meta-actions (SPSM) that brings the meta-module from its current state to the state being evaluated. The following inputs are given to the ANN for each state:

- Number of meta-actions in the SPSM.
- Number of common meta-actions between the current SPSM and the previous SPSM. The previous SPSM is the latest sequence of meta-actions from which the meta-module performed the first meta-action.
- Shortest distance to another meta-module, measured from a state along the SPSM.
- Reduction in distance between the meta-module and its nearest attraction-point, if it moves to the state being evaluated.

The state, which is being evaluated, is assigned a fitness value from the output of the ANN. The state in the reachable-space subset which has the highest fitness value is selected. And the first meta-action in the corresponding SPSM is then performed by the meta-module. Since the structure of modules is dynamic and information in the map may be incomplete it happens that the performance of a meta-action fails. The meta-module then recovers as good as possible, e.g. if a rotation fails because of collision the rotation will be inverted and meta-action cancelled.

D. Stopping of Meta-module

Each time the meta-module has performed a meta-action it decides if it is time to stop or perform another meta-action. An ANN with 5 neurons in input-layer, 3 neurons in hidden-layer and 1 neuron in output-layer makes this decision based on the following inputs:

- Biggest known possible reduction in the distance between the meta-module and its nearest attraction-point.
- Distance to nearest attraction-point from current state of the meta-module.
- Number of possible connections between the meta-module and its passive neighbours modules.
- Reduction in distance to nearest attraction-point over the last five meta-actions.
- Number of cancelled meta-actions (e.g. because of collision) in the lifetime of the meta-module.

If the output of the ANN is greater than 0.5 the meta-module will stop moving and connect to all passive neighbour modules.

VII. Evolution of Artificial Neural Network Controller

Evolution is chosen to optimize the value of the ANN's weights, since there is no obvious way of training the network and since evolution may be good to exploit cooperation between meta-modules. The actions of one meta-module may affect other meta-modules in ways which are difficult to analyze and harder to exploit.

A. Encoding of Artificial Neural Networks

The topologies of the networks are fixed, only the weights are optimized by means of evolution. The genome of each individual is the 50 weights which are directly encoded as floating points values. Initially the weights have random values between -0.5 and 0.5.

B. Genetic Algorithm

A simple genetic algorithm is used. Each generation consist of 100 individuals. The individuals in each generation are evaluated and their fitness calculated. The 3 fittest individuals are used as elites and directly copied to the new generation. A child has two parents randomly selected from the group of the 25 fittest individuals. The child is produced by using a randomized 12-point crossover and a mutation rate of 5%. When mutating a gene it is with equal likelihood replaced with a new random value or a small random value added to or subtracted from the gene.

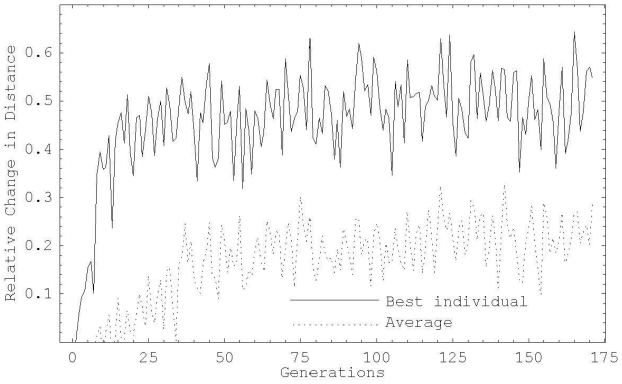


Fig. 5. The graphs show the average and highest fitness for each generation, when evolving the weights of the artificial neural network controller for the ATRON modules.

C. Fitness Evaluation

To evaluate the individuals they perform two randomly generated tasks. A task is to shape-change a random structure of 50 modules into another random structure specified by 50 inhibiting attraction-points, placed within the ATRON lattice. The individuals in a generation are all evaluated on the same random tasks, but from generation to generation new random tasks are generated. The initial and target shape of the modules are not far apart, on average 45% of the modules are initially placed at an attraction-point. A task is terminated if there within 300 timesteps has been no decrease in Euclidian distance between the modules and the attraction-points. In the simulator a 90 degree rotation takes 10 timesteps. The fitness of an individual is calculated as the average fitness from each of the two tasks. The fitness from a task is calculated as the relative change in distance between the structure of modules and the attraction-points: $fitness = (D_{start} - D_{end})/D_{start}$. Distance between the structure of modules and the attraction-points is measured as the sum of Euclidian distances between a module and its nearest attraction-point.

A number of evolutionary experiments were performed before reaching the details described above. The final controller, used for experiments in this work, was obtained from a evolutionary run for which the fitness graph is shown in figure 5. The noise in the fitness evaluation indicates that some tasks are harder to solve than other.

VIII. Experiments

In this section we demonstrate how shape-change and self-repair behaviours may be achieved using attraction-points and the evolved artificial neural network controller. The purposes of the experiments are to: 1) validate the control strategy presented in this paper, 2) demonstrate possible future applications of self-reconfigurable robots.

A. Experiment: Scalability

The ANN controller was evolved using tasks containing 50 modules. To investigate how the controller scaled to shape-changing structures of more modules we measured the controller's performance as it performed a series of

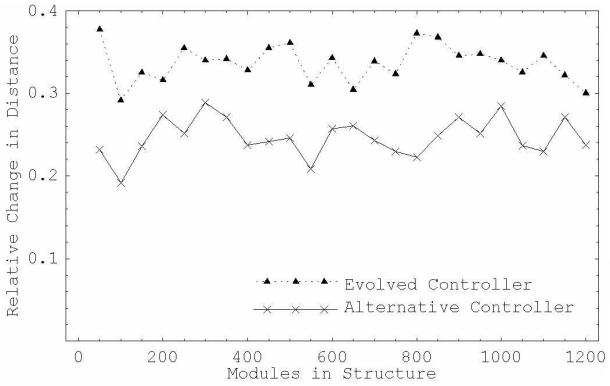


Fig. 6. The graphs show how the relative change in distance of an evolved and an alternative hand-coded controller scale from 50 to 1200 modules in the structure. Each point in the graphs is calculated as the average of 10 tasks. Calculated from the entire interval, the evolved controller has a 95% confidence interval for the mean of 0.325 to 0.345 and a standard deviation of 0.0787. The alternative controller has a 95% confidence interval for the mean of 0.237 to 0.260 and a standard deviation of 0.0875. The evolved controller performs significantly better than the alternative controller.

tasks. Tasks were of the same random type as used when evolving the controller, but the number of attraction-points and modules were varied from 50 to 1200 in steps of 50 modules. The graph in figure 6 indicate that the relative change in distance, when performing a task, does not decline as a function of the number of modules in the structure. Therefore the experiment indicates that the evolved controller scales up to at least 1200 modules.

For comparison the equivalent graph of an alternative controller is also shown in figure 6. This alternative controller is hand-coded based on its reachable-space subset and does not use ANN's. The alternative controller:

- Emerge, if it is able to reduce its distance to the nearest attraction-point.
- Move, along the shortest path of meta-actions, towards the state that minimize its distance to an attraction-point.
- Stop, if it no longer is able to reduce its distance to the nearest attraction-point.

The experiments indicate that the evolved controller performs significantly better than the alternative controller.

B. Experiment: Support Unstable Roof

In earth quake or cave environments it might be desirable to have systems which can support an unstable roof. In this experiment a random structure of 500 modules initially lay on a floor. A roof is positioned at the height of 26 modules above the floor. The target functionality is the same as that of a pillar. To achieve this functionality 117 inhibiting attraction-points are placed in a column shape. As the robot change-shape upwards the attraction-points of lower positions will be inhibited by the modules at their positions. The result of this simulated roof supporting experiment is shown in figure 1.

C. Experiment: Bridge Gap

In a range of scenarios it may be useful to have a system which can build a bridge across a gap. In this

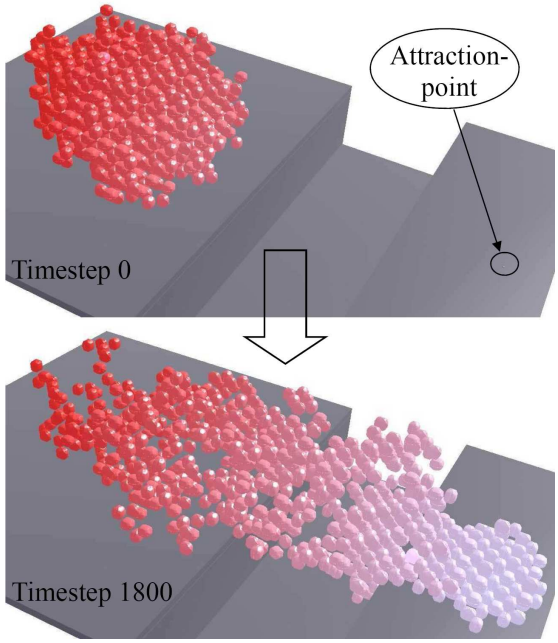


Fig. 7. A structure of 1000 ATRON modules build a bridge across a gap. A single non-inhibiting attraction-point is used to guide the modules shape-change.

experiment 1000 ATRON modules are initially placed in a random structure. A single non-inhibiting attraction-point is placed as shown in figure 7. The shape of the ATRON robot stretches towards the attraction-point, effectively building a bridge across the gap. The ATRON simulator does not support physics. During this experiment long thin arms of ATRON modules are build, which most likely would break off in a real-world gravity environment. This problem could be reduced by adding more attraction-points to guide the shape-change.

D. Experiment: Self-Repairing Bone

A future miniaturization of modules would open up for new possible applications, e.g. smart material which could self-repair. This experiment demonstrate the use of miniature ATRON modules in a self-repair scenario. Initially, using a CAD model, 3426 ATRON modules are assembled in the form of a bone, see figure 8. A total of 1663 inhibiting attraction-points are placed at the same positions as ATRON modules which is connected to eight neighbours. This leaves the surface of the bone free of attraction-points. At timestep 20, 114 modules are removed which damage the strength of the bone. Since the removed modules no longer inhibit the attraction-points this triggers the emergence of meta-modules. After 1000 timesteps the modules have rearranged themselves, the bone is self-repaired and its strength largely recovered.

IX. Conclusion and Future Work

In this paper we have described a distributed shape-change and self-repair control strategy for the ATRON robot. A key ingredient in the control strategy is the modules' continuous calculation of a subset of their reachable-

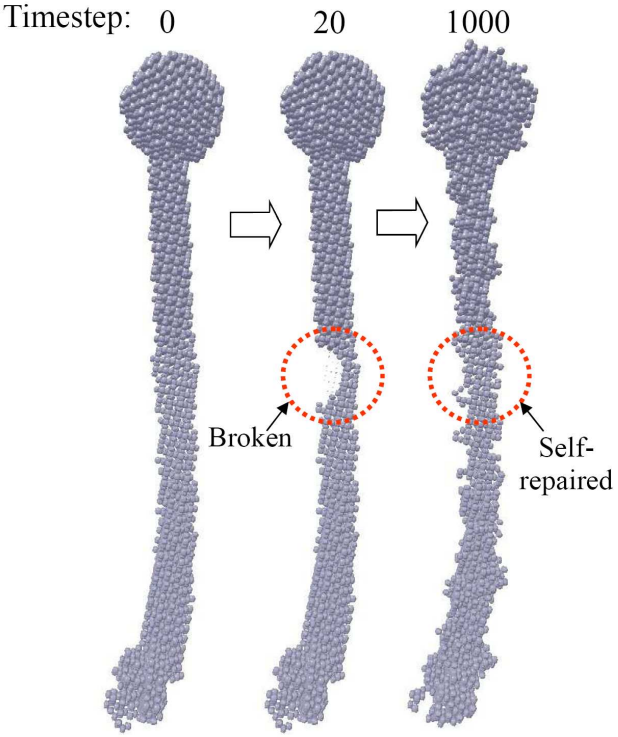


Fig. 8. Self-repair of a bone build from 3426 ATRON modules: At timestep 0, there is no activity. At timestep 20, the bone breaks. At timestep 1000, modules have rearranged themselves to self-repair the bone.

space graph. On the basis of this graph we evolved an artificial neural network controller for the modules. The controller controls the emergence, movement and stopping of meta-modules composed of three ATRON modules. Tasks of the robot are specified using attraction-points which trigger meta-modules to emerge and move towards them. The combination of meta-modules and evolved artificial neural network controller are shown to be able to deal with the difficult motion constraints of the ATRON modules. The control strategy allows the robot to change its shape to meet some desired functionality. In simulation we have verified that the control strategy scales to shape-change and self-repair scenarios with several thousands of modules in the robot. Future work includes the transference of the evolved controller to the physical ATRON modules.

References

[1] L. J. Guibas A. Nguyen and M. Yim. Controlled module density helps reconfiguration planning. In *Proc. 4th International Workshop on Algorithmic Foundations of Robotics*, 2000.

[2] F. H. Bennett and E. G. Rieffel. Design of decentralized controllers for self-reconfigurable modular robots using genetic programming. In *Proceedings, Second NASA/DoD Workshop on Evolvable Hardware*, 2000.

[3] H. Bojinov, A. Casal, and T. Hogg. Multiagent control of self-reconfigurable robots. In *Fourth International Conference on MultiAgent Systems (ICMAS)*, pages 143–150, 2000.

[4] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Cellular automata for decentralized control of self-reconfigurable robots. In *Workshop on Self-reconfigurable Robots, IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[5] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for a class of self-reconfigurable robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[6] D. J. Christensen, E. H. Østergaard, and H. H. Lund. Metamodule control for the ATRON self-reconfigurable robotic system. In *Proceedings of the The 8th Conference on Intelligent Autonomous Systems (IAS-8)*, pages 685–692, Amsterdam, 2004.

[7] R. Fitch, D. Rus, and M. Vona. A basis for self-repair using crystalline modules. In *Proceedings, Intelligent Autonomous Systems (IAS-6)*, Venice, Italy, 2000.

[8] M. W. Jorgensen, E. H. Østergaard, and H. H. Lund. Modular atron: Modules for a self-reconfigurable robot. In *In proceedings of IEEE/RSJ International Conference on Robots and Systems, (IROS)*, pages 2068–2073, Sendai, Japan, 2004.

[9] A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, S. Murata, and S. Kokaji. Automatic locomotion pattern generation for modular robots. In *ICRA*, pages 714–720. IEEE, 2003.

[10] K. Kotay and D. Rus. Algorithms for self-reconfiguring molecule motion planning. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[11] J. Kubica, A. Casal, and T. Hogg. Complex behaviors from local rules in modular self-reconfigurable robots. In *Proceedings of the IEEE Int. Conference on Robotics and Automation(ICRA)*, volume 1, pages 360–367, May 2001.

[12] C. McGray and D. Rus. Self-reconfigurable molecule robots as 3d metamorphic robots. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, 1998.

[13] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, and S. Kokaji. A 3-d self-reconfigurable structure. In *Proceedings, IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[14] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-TRAN: Self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4):431–441, 2002.

[15] E. H. Østergaard and H. H. Lund. Evolving control for modular robotic units. In *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 886–892, 2003.

[16] E. H. Østergaard and H H. Lund. Distributed cluster walk for the atron self-reconfigurable robot. In *In Proceedings of the The 8th Conference on Intelligent Autonomous Systems (IAS-8)*, pages 291–298, Amsterdam, Holland, 2004.

[17] K. C. Prevas, C. Unsal, M. O. Efe, and P. K. Khosla. A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2002.

[18] D. Rus and M. Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, Jan. 2001.

[19] K. Stoy and R. Nagpal. Self-repair through scale independent self-reconfiguration. In *In proceedings of IEEE/RSJ International Conference on Robots and Systems, (IROS)*, 2004.

[20] K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, and S. Kokaji. A self-assembly and self-repair method for a distributed mechanical system. *IEEE Transactions on Robotics and Automation*, 15(6):1035–1045, Dec 1999.

[21] C. Unsal and P. Khosla. A multi-layered planner for self-reconfiguration of a uniform group of i-cube modules. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2001.

[22] C. Unsal, H. Kiliccote, and P. K. Khosla. I(ces)-cubes: A modular self-reconfigurable bipartite robotic system. In G.T. McKee and P.S. Schenker, editors, *Proceedings, SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, pages 258–269. SPIE, 1999.

[23] S. Vassilvitskii, J. Kubica, E. Rieffel, J. Suh, and M. Yim. On the general reconfiguration problem for expanding cube style modular robots. In *Proceedings of the 2002 IEEE Int. Conference on Robotics and Automation (ICRA)*, 2002.

[24] E. Yoshida, S. Murata, and A. Kamimura. A motion planning method for a self-reconfigurable modular robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.