Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Comparison of A* and RRT-Connect Motion Planning Techniques for Self-Reconfiguration Planning

David Brandt
The Adaptronics Group
The Maersk Mc-Kinney Moller Institute for Production Technology
University of Southern Denmark
david.brandt@mip.sdu.dk

***Abstract*— This paper presents a comparison between two different algorithms for self-reconfiguration planning on modular self-reconfigurable robots. The two algorithms are A* and RRT-Connect. A* has often been used for self-reconfiguration planning on different systems, but so far no work has been done to show that A* is an optimal or even a good choice for this task. The RRT-Connect algorithm is adopted from traditional robot motion planning and is easily adapted to self-reconfiguration planning. The results show that RRT-Connect is significantly faster than A* for difficult self-reconfiguration problems. In most cases RRT-Connect generates longer paths that A* it is, however, possible to optimize the paths from RRT-Connect by a simple and efficient method such that the difference is negligible.**

***Index Terms*— self-reconfiguration, planning**

I. Introduction

Within the field of traditional robot motion planning efficient methods exist for path planning. These methods have typically been tested on different setups containing one or more industrial robots. In this domain the path planning problem can be described as the problem of finding a path from an initial configuration to a goal configuration which must be fully contained in the collision free part of the configuration space $\mathcal{C}_{free}$.

For lattice-based modular self-reconfigurable robots the problem is quite similar. Here the problem is to find a path from an initial configuration to a goal configuration which must be fully contained in the valid part of the configuration space $\mathcal{C}_{valid}$. Where $\mathcal{C}_{valid}$ is a subset of $\mathcal{C}_{free}$ containing the configurations neither leading to collisions of the modules or with the environment nor to disconnection of the structure. One additional difference between the two problems is that in the classical problem the configuration space is continuous and in the case of lattice-based modular self-reconfigurable robots the configuration space is discrete.

The planning problem for self-reconfigurable robots is, however, much more difficult than for traditional robots which is manly due to two different factors. First, the number of DOFs in a self-reconfigurable robot is often much higher than in a classical robot. The second problem is that the motion constraints on a self-reconfigurable robot is much more complex, which makes it difficult to devise a suitable metric for measuring the distance between two configurations.

This paper will present our work on adapting the classical robot motion planning method RRT-Connect by Kuffner et al. [1] for use on the ATRON lattice-based modular self-reconfigurable robot system and a comparison with the A* self-reconfiguration planning method.

II. Related Work

Robot motion planning for classical robots in known and static environments is a well studied subject containing a large set of different methods with different advantages and disadvantages. One of the most important types of planners is the probabilistic road map planners (PRM-planners) which first builds a road map of the configuration space and then later uses this road map for handling queries. The road map approach is obviously good for handling multiple queries in the same environment since the road map can be reused with ease. The initial work on PRM-planners was done by Kavraki et al. in [2].

Another method for path planning is RRT-Connect by Kuffner et al. [1]. This method works by dynamically building two trees in the configuration space from the initial configuration and the goal configuration and then tries to connect these two trees. The RRT-Connect planner is known to provide good results for single query tasks in both open and maze-like environments.

In [3] Hsu et al. present a single query PRM-planner. Much like RRT-Connect the planner is based on randomly expanding two trees in the configuration space. The main differences are the strategies used in the generation of new nodes and the connecting to these nodes.

In the modular self-reconfigurable robot domain planning has previously been used on a number of different systems. In [4], Pamecha et al. present a method for planning self-reconfiguration on a class of modular self-reconfigurable robots which they call *metamorphic* robots. The difference between metamorphic robots and general self-reconfigurable robots is that in metamorphic robots a module has the ability to climb on the surface of the cluster of modules without help from other modules or from the environment. This constraint on the design of the modules makes it somewhat easier to plan the self-reconfiguration. They use simulated annealing where the energy function is the distance between the two configurations.

In [5], Ünsal and Khosla present a method for motion planning for the I-Cubes system. The planning method is

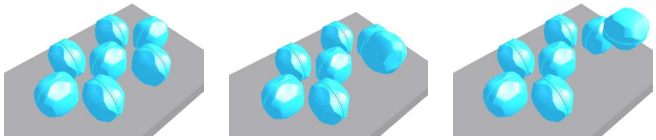

Fig. 1. A small picture sequence illustrating the basic principles of how one ATRON module is dependent upon the others in order to move within the cluster. The connections between modules are not shown.

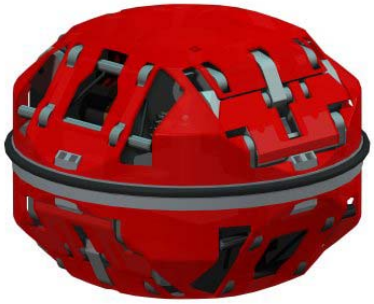

Fig. 2. Picture of an ATRON module. The modules have on board power, computation and local communication. The parts are mostly made from aluminium.

based on the A* algorithm and uses the Manhattan metric. A three-layer planner for the I-Cubes system is presented in [6] by Prevas et al. The introduction of metacubes aids the planning such that the planning at the highest level plans the movement of metacubes.

In [7] Yoshida et al. present a method for planning movements of a cluster of M-Tran modules with a specific structure. The technique is based on a two-layer approach consisting of a global planner for planning the cluster flow and a motion scheme selector for planning the motions of each single module. The motion scheme selector is based on a database of possible paths for the module.

In [8] Casal and Yim present a divide-and-conquer method for self-reconfiguration planning for chain type systems. The method is called Hierarchical Substructure Decomposition and the basic workings of the method is that the structures[1] are decomposed into substructures based on predefined classes such as $chains$ and $loops$. When the decomposition is completed the reconfiguration is achieved by utilizing small sequences of actions among the substructures.

III. ATRON

The robot platform used for this work is the ATRON system which is a lattice-based modular self-reconfigurable robot system consisting of heterogeneous modules. The basic module design is based on two hemispheres connected by a rotational joint. Furthermore, each hemisphere is equipped with four connectors, two male ones and two female ones; the male/female connector design is chosen for mechanical reasons. Figure 2 shows a single ATRON module. Figure 1 shows how a module can move another module by connecting to it and rotating. The design is such that one module is capable of lifting two other modules. A complete description of the module design can be found in [9].

[1]The term *structure* in the paper by Casal and Yim is analogue to our use of the term *configuration*

IV. ATRON Simulator

In the planning algorithms a major part of the workload is testing whether a given action is legal in a given configuration, and the computation of the resulting configuration. To achieve this task a simulator for the ATRON system is used. The different timing issues are simulated by running the simulator in time steps, and at every time step the controllers of the modules are simulated one at a time in random order, described as the D1 activation scheme in [10]. The simulator contains routines for collision testing which handles both module/module collisions and module/environment collisions. The collision test assumes that only one module is acting at a time, which is not a problem in this work since the planning algorithms are already based on this assumption. The planning algorithms are described in section V. The simulator contains no simulation of real-world physics such as friction, stability etc. Screen shots from the simulator can be seen in figures 1 and 4.

V. Planning Algorithms

In this work two different approaches to self-reconfiguration planning will be compared. The methods are the A* algorithm and the RRT-Connect method by Kuffner et al. [1]. The two methods will be evaluated on the ATRON self-reconfigurable robot system which is described in section III and in [9]. To avoid a combinatorial explosion of the number of possible actions that can be taken in a given configuration some restrictions have to be made. We have chosen to allow only a single module to act at any given time. This reduces the number of possible actions significantly. The main disadvantage of this approach is the reduced speed of the system and the need to synchronize the movements. The number of useful actions that require more than one module to act at a time is limited, so the basic capabilities of the system do not change significantly due to this restriction.

A. RRT-Connect Planning for Self-Reconfigurable Robots

The RRT-Connect motion planning method was developed by Kuffner and LaValle in [1]. This method is chosen because it is known to provide good results for single-query path planning in the field of traditional robot motion planning, which is what we are interested in in this context.

The method is randomized and does not provide any guarantees regarding the quality of the solutions found or the time required to find them. However, the method is probabilistic complete and it is guarenteed to find a solution if such exists given a sufficient amount of time.

The basic idea is quite simple: starting with only the initial configuration and the goal configuration, two trees are built

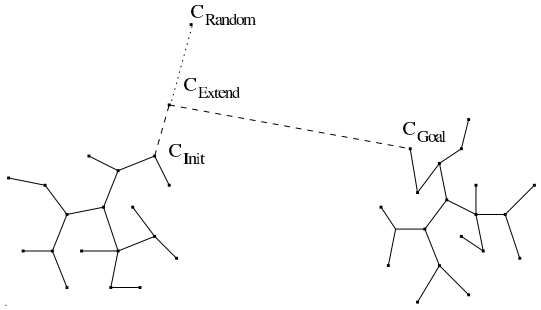

Fig. 3. The basic idea behind RRT-Connect. First a random configuration C_{Random} is generated, then a node close to it is selected as origin node C_{Init}. From C_{Init} a small step is made towards C_{Random} ending in C_{Extend}. Finally, it is attempted to connect to a close node in the other three C_{Goal} from C_{Extend}.

in the configuration space, and while the trees are being constructed, repeated attempts are made to connect the trees. If a connection is achieved, then the path is easily found by traversing the resulting tree from the initial configuration to the goal configuration. The trees are called *Rapidly exploring Random Trees* hence the name RRT-Connect.

The algorithm consists of four basic steps:

1) Generate a random configuration.
2) Extend the tree from the node closest to the generated configuration and towards it for one step.
3) Try to connect the last generated node to the other tree.
4) Swap the two trees and repeat until a connection is made.

The workings of RRT-Connect is illustrated in figure 3.

In order to implement the algorithm the first three steps have to be considered.

In the first step, a random configuration is generated; this is done by first generating a random position for one module within some area of $\mathcal{R}^3$, then adding modules incrementally by choosing randomly unblocked connectors on the cluster to connect to. This way of generating random configurations ensures that the configuration is connected, but there is no guarantee that a path to the configuration exists.

In step two, one of the trees should be extended towards the random configuration. One obvious solution to this problem is to try all actions from the given configuration and then choose the legal one that reduces the distance to the goal the most. The metric for measuring the distance to the goal is described in section VI. However, this approach is not good at handling the many small local minima of the distance metric. Therefore, a slightly different and unfortunately computationally expensive solution is used. Instead of only looking one action ahead all possibilities of sequences up to three legal actions are considered. This approach ensures that the planner does not get stuck in small local minima, which in turn makes the global planning much easier. The choice of precisely three actions is based on empirical tests leading to believe that this is a good choice for the ATRON system. For different module morphology the optimal number of actions might very well be different. Since we do not want to get to the random configuration, but only move towards it, we perform only one step.

The last step to be considered is step three. This step is implemented much like step two. First, a node from the other tree is chosen as the goal. This choice is at random, but with a higher probability for close nodes. Then we extend towards the goal node, like in step two, but this time we repeat until no improving action sequence can be found or until the goal is reached.

B. A* Planning for Self-Reconfigurable Robots

The A* algorithm developed by Hart et al. in [11] is known as an algorithm for computing shortest path in a graph, but it can easily be adapted to self-reconfiguration planning. The specific algorithm used in this work is basically a greedy graph search: the most promising nodes are branched first until a path from the initial configuration to the goal configuration is found.

The main difference between A* and RRT-Connect is that A* is not randomized. Furthermore, A* is resolution complete and will find the optimal solution if provided with a suitable metric for the problem at hand. Unfortunately, such a metric is not known for the self-reconfiguration problem and, therefore, we have no guarantees for the quality of the solutions found, but it is expected that the solutions found by A* is of good quality. The metric used is described in section VI.

A sketch of the algorithm looks like this:

1) Create a node containing the initial configuration and put it in the *Open* set.
2) Repeat steps 3–6 while *Open* is not empty and a path has not been found.
3) Get the node from *Open* that is closest to the goal configuration.
4) If the distance to the goal configuration is zero, the path has been found.
5) Generate all successors of this node and put them in *Open* if they are not in *Open* or *Closed*.
6) Put the branched node in *Closed*.

When generating the successors of a node, we generate all configurations reachable by a sequence of three legal actions to avoid small local minima like in the extend step in RRT-Connect.

VI. ATRON Configuration Metric

In order to use planning on self-reconfigurable robots it is necessary to have some way of measuring the distance between two configurations. The optimal metric would measure the minimum number of atomic actions needed to change from one configuration to another. Using this metric planning would be trivial since a simple steepest decent would solve the problem optimally. However, there is currently no method for calculating this distance between two configurations that runs in less than exponential time. Furthermore, the calculation of this optimal metric is probably NP-complete since it would also solve the optimal reconfiguration problem that bears the

hallmarks of a NP-complete problem, but it remains to be formally proven.

Since the optimal metric cannot be used, it is necessary to find a suitable alternative that is as close to the optimal as possible while still being fast to compute. The metric we used is basically the same as Pamecha et al. use in [4]. The metric is based on the optimal assignment which can be computed in $O(n^3)$ by the Hungarian method developed by Kuhn [12].

The metric proposed in this paper consists of two main parts.

The first part is a simple metric for calculating the distance between two single ATRON modules. This metric is based on three properties of the two modules being compared. The positions, the orientations and the connectivity. The complete module-to-module distance function looks like this:

$$dist(a_1, a_2) = max(|a_{1_x}, a_{2_x}|, |a_{1_y}, a_{2_y}|, |a_{1_z}, a_{2_z}|) + \frac{differentOri(a_1,a_2)}{2} + \frac{differentConnectors(a_1,a_2)}{16}$$

As distance measure for the positions of two modules we use the maximum axis aligned distance with the axes aligned according to the lattice structure of the configuration.

The function *differentOri* returns 1 if the orientations of the modules are different and 0 if they are equal. Two module orientations are considered equal if the two modules have male/female connectors in the same directions.

The function *differentConnectors* returns the number of connectors that are in different states on the two modules i.e. the connector on one module is connected and the same connector on the other modules is not. This is a number in the range from 0 to 8.

Finally the three properties are weighted such that the position is allways the most important property, then follows the orientation and finally the connectivity as the least important property. The idea behind the priority of the properties is that it seems to be a good idea to first get the modules to the correct positions, then get them oriented correctly and finally get them connected in the correct manner.

The distance caluclated by the complete metric is then the sum of module-to-module distances, where each module in the first configuration is assigned to a module in the second configuration such that the sum of module-to-module distances is minimal. This is the optimal assignment problem which is solved using the Hungarian method [12].

VII. Experiments

To test the two different planning methods a set of 600 planning problems were generated. Each planning problem consists of two configurations, the goal is to find a sequence of atomic actions transforming the initial configuration to the goal configuration.

The generation of the problem pairs was done by starting with the initial configuration and then performing random actions until the distance between the generated configuration and the initial configuration exceeded a certain threshold. In order to generate problems with different levels of difficulty the threshold was set to values in the range from 2–30 in steps of 2. In each step 40 configuration pairs were generated.

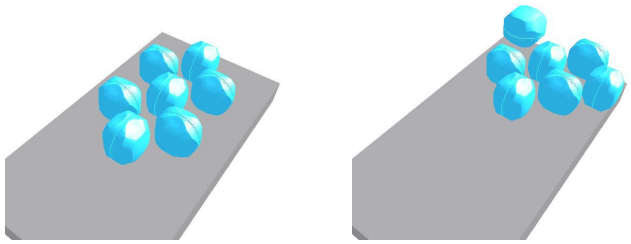

Fig. 4. Screen shots of two configurations from the test set. The configurations in the test set are generated by performing random actions. By doing this, we ensure the existence of a path between the two configurations. The test set consists of a total of 600 test pairs.

Figure 4 shows an example of a configuration pair from one of the tests.

The criterion of success when running the two algorithms was that they were able to find a path from the initial configuration to the goal configuration while examining less than 10^7 configurations. The choice of the specific limit is based on the time required to run the tests. This limit leads to a maximum planning time of approx. 30 minutes for each problem on a 1.8GHz Intel Pentium IV. If no path was found within the limit then the trial was a $failure$. The number of failures and the number of examined configurations was then recorded for both algorithms during one run on the entire test set.

To test the quality of the generated paths a second experiment was conducted. 20 random configuration pairs were generated with distances varying from 2–20. Each pair was then evaluated by hand in order to find a near optimal solution. The two planners were then used to solve the 20 problems and the number of actions used was compared to the solutions found by hand.

VIII. Results

The failure rate of the two planning methods is shown in the left part of figure 5.

The failure rate of A* is generally increasing as the distance between the two configurations increases, but it seems that problems of distance 14 are easier to solve than problems of distances 10 and 12 and problems of distance 24 are easier than problems of distance 22.

For RRT-Connect problems with distances of up to 20 are solved without failures, and for problems with distance 30 the failure rate reaches approximately 50%.

The difference between the performance of A* and RRT-Connect is statistically significant for problems with distances from 10–12 and from 16–30. The used statistical test is the likelihood ratio test with a significance level of 97.5%.

The right part of figure 5 shows the average number of examined configurations for the RRT-Connect planner for the problems with a distance of less than or equal to 20, that is the

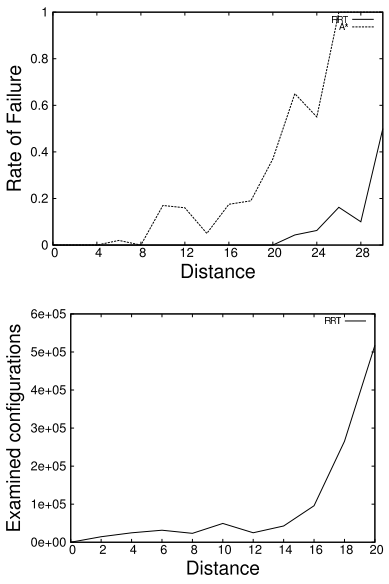

Fig. 5. **Top:** The rate of failure for A* and RRT-Connect for problems with different distances between the initial configuration and the goal configuration. When the distance reaches 26 the A* algorithm is unable to solve even a single one of the 20 test problems. RRT-Connect, however, is still able to solve approximately 50% of the test problems when the distance reaches 30. **Bottom:** The average number of configurations examined by the RRT-Connect planner before a path to the goal is found. The resemblance with an exponential growth is quite obvious. In both figures the distance is calculated by the ATRON configuration metric as described in section VI.

problems where no failures occurred. Again it seems that the planning problems with distances 8 are easier than expected, when compared to the problems with smaller distances, but the difference is not as apparent as in the previous test.

In general, the results show that the RRT-Connect planner becomes much more efficient than A* when the planning problems become complex. Furthermore, it can be seen that even though RRT-Connect performs much better than A* there is still an exponential growth in the work load as the planning problems become more complex.

The results form the second test are shown in the following table:

Best known	A*	RRT	RRT optimized
5.71	7.05	13.82	8.65

The first column contains the average length of the best known solutions to the 20 problems. The second column contains the average length of the solutions found by A*. The third contains the average length of the solutions from RRT-Connect. The final column contains the average length of the results from RRT-Connect with a simple post processing applied. The post processing simply finds loops or near loops in the path and removes them. A loop is identified by the fact that the same configuration is visited more than once in a path. A near loop is a situation where two not neighbouring configurations in the path can be connected by a single action. It was also attempted to optimize the paths from A* but this did not improve the quality. In general, the time used for the post processing is negligible compared to the time used for the actual planning.

IX. Discussion

From the experiments it is clear that the RRT-Connect method is better at solving self-reconfiguration planning problems than A*; in the case of planning problems with a distance between the two configurations of 20, A* had a failure rate of approximately 40% whereas RRT-Connect did not fail in any trials.

The main reason for the quite big difference in performance is probably the greedy approach used in A*. If there is a deep local minima in the search space, then A* will have to make an exhaustive search of the configurations in the local minima, before proceeding to configurations further away from the goal, but outside the local minima. This disadvantage consumes a lot of computational time and the presence of such a minima gets more likely as the distance between the initial configuration and the goal configuration is increased. The randomized nature of RRT-Connect makes this method less sensitive to such minima.

Even though RRT-Connect performs much better than A* it is not feasible to use it for large planning problems. It is still necessary to use some mechanisms in order to make the planning problem easier, for instance by using meta-modules as in [6]. On the ATRON system meta-modules have been used in [13] but not in connection with explicit planning. Another approach for making the problem easier for the planner is to sub-divide the desired global behaviour into smaller sub-behaviours which easily can be planned. Afterwards the sub-behaviours are combined into the global behaviour [14].

The quality of the paths found by RRT-Connect showed to be much poorer than the quality of the solutions found by A*, which was allmost as good as the ones found by hand. When the simple path-optimizing procedure was applied to the paths, the lengths of the paths from RRT-Connect were reduced greatly, such that the average difference in length was only 1.6 actions. If the lengths of the paths are very important in the application at hand, it might be worth the extra computational time to use A* and not RRT-Connect, but for most applications RRT-Connect is the best choice.

One of the very important aspects of planning is the choice of metric. If an optimal metric could be devised, then the planning problem would be a simple steepest decent. But as discussed in section VI the calculation of this metric is probably NP-complete even though this remains to be proven. The metric presented in this paper is a reasonable measure of the difficulty of the planning problem, but not optimal. In the left part of figure 5 it seems that planning problems with a distance of 14 between the initial configuration and the goal configuration are easier than problems with distances of 10 or 12. But, in general, the problems tend to get more and

more difficult to solve as the distance grows.

The implemented methods were also tested on a simulator for the M-TRAN system, but here the results were much poorer. The planners could only solve very simple planning problems involving only a few actions. The reason for this is, probably, the very complex motion constraints of the M-TRAN system where especially the rotation of modules is very difficult and often impossible. This leads to a poor quality of the used metric since it assumes that the rotation of a module is easier than getting it to the correct position, which is not true for the M-TRAN system. So it seems that it is necessary to devise a specific metric for each type of modules in order for the planning to work efficiently.

If we look at the use of planning on self-reconfigurable robots in general, there is a number of pros and cons of this approach when compared to a distributed reactive approach. One of the main advantages is that it is relatively easy to handle different aspects of the environment such as the stability of the structure, the maximum allowed forces on the connectors etc. by using a simulator, which includes these aspects. How to handle these aspects in a distributed reactive controller remains to be investigated.

There are two main disadvantages. The obvious one is the computational explosion that makes planning for problems of just a reasonable size infeasible; this problem might be bypassed as discussed earlier. The second problem is that the planning approach is of little use for modular self-reconfigurable robots since it is desirable to have distributed control and the planning approach is inherently centralized. The solution to this problem could be to somehow generate distributed controllers based on the results from the planning algorithms. So a question is how to use these generated action sequences to generate controllers for the modules that work in a distributed manner.

One possibility for the generation of controllers is to use some variation of rule-based controllers as the ones used in [14] and [15]. The rule sets in the rule-based controllers could then be generated from the action sequences from the planner. In order to make the planning easier the global behaviour could be sub-divided and the sub-behaviours could be recombined as done in [14] where Brandt and Østergaard present a method for combining sub-behaviours for modular robots into a global behaviour.

X. Conclusion

In this paper we show that it is possible to improve the performance of planning approaches for self-reconfiguration planning for the modular self-reconfigurable robot ATRON by adopting some of the techniques used in traditional robot motion planning. More precisely, two experiments have been conducted. One uses the A* method for planning. The second experiment used the RRT-Connect method developed by Kuffner and LaValle. The results show that the RRT-Connect planner was significantly faster than the A* algorithm when the planning problem is sufficiently difficult. The quality of the solutions found by A* was close to the quality of the best known solutions. The quality of the solutions found by RRT-Connect was poorer, but by applying a simple optimization procedure the quality was improved to levels close to that of A*.

Acknowledgment

The ATRON and M-TRAN simulators used in this work were developed by Esben H. Østergaard.

References

[1] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proceedings of the 2000 IEEE Internatinal Conference on Robotics and Automation (ICRA2000)*. IEEE Press, 2000.

[2] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, August 1996.

[3] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *International Journal of Computational Geometry & Applications*, vol. 9, no. 4-5, pp. 495–512, 1999.

[4] A. Pamecha, I. Ebert-Uphoff, and G. S. Chirikjian, "Useful metrics for modular robot motion planning," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 4, pp. 531–545, 1997.

[5] C. Ünsal and P. K. Khosla, "Solutions for 3-d self-reconfiguration in a modular robotic system: Implementation and motion planning," in *Proceedings of SPIE, Sensor Fusion and Decentralized Control in Robotic Systems III*, 2000.

[6] K. C. Prevas, C. Ünsal, M. O. Efe, and P. K. Khosla, "A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA2002)*. IEEE Press, 2002, pp. 787–792.

[7] E. Yoshida, S. Murata, A. Kamimura, K. Tomita, H. Kurokawa, and S. Kokaji, "A motion planning method for a self-reconfigurable modular robot," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS2001)*. IEEE Press, 2001, pp. 606–612.

[8] A. Casal and M. Yim, "Self-reconfiguration planning for a class of modular robots," in *SPIE Proceedings*, G. T. McKee and P. S. Schenker, Eds., vol. 3839, 1999, pp. 246–257.

[9] M. W. Jørgensen, E. H. Østergaard, and H. H. Lund, "Modular ATRON: Modules for a self-reconfigurable robot," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2004)*. IEEE Press, 2004, pp. 2068–2073.

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

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths in graphs," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[12] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[13] 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)*, Amsterdam, Holland, March 2004, pp. 685–692.

[14] D. Brandt and E. H. Østergaard, "Behaviour subdivision and generalization of rules in rule based control of the ATRON self-reconfigurable robot," in *Proceeding of International Symposium on Robotics and Automation (ISRA2004)*. Secretaria de Educación Pública, 2004, pp. 67–74.

[15] E. H. Østergaard and H. H. Lund, "Distributed cluster walk for the ATRON self-reconfigurable robot," in *Proceedings of the The 8th Conference on Intelligent Autonomous Systems (IAS-8)*, Amsterdam, Holland, March 2004, pp. 291–298.