Navigating modular robots in the face of heuristic depressions

Kazuo Miyashita* and Shigeru Kolaji
National Institute of Advanced Industrial Science and Technology
1-2-1, Namiki, Tsukuba, Ibaraki 305-8564, Japan

Abstract. A modular robotic system is composed of a number of modules, each of which can change its position relative to its neighboring modules under certain physical constraints. The modular robotic system can move itself flexibly by repeated motions of its component modules. However, a huge size of possible combinations of subsequent module motions and tight physical constraints among them cause difficulty in making an appropriate plan for the modular robotic system to reach a designated goal position, especially when it is located in unfamiliar environments with obstacles. Heuristic search methods fail to find a plan efficiently due to poor cost estimation used in the search (i.e., heuristic depression problem). In this paper, we propose a method for navigating the modular robotic system by extending a real-time heuristic search algorithm to overcome the heuristic depression problem. The experimental results show that the proposed method is effective for navigating the modular robots in several problem settings.

Key words: modular robots, real-time heuristic search, heuristic depression

1 Introduction

A modular robotic system consists of independently controlled modules, which can connect, disconnect and walk over/through the neighboring modules. Dynamic mobility of component modules results in self reconfiguration and locomotion capabilities of the modular robotic system. Because of its modularity, redundancy and homogeneity, the modular robotic system is (1) adaptive to dynamically changing environment, (2) robust against malfunction and (3) suitable for mass production.

There have been many research activities in developing modular robotic systems [3,6,29,4,12] and several types of the hardware systems have been proposed, and at the same time, computational intractability in planning and controlling motions of the modular robotic systems has been pointed out and tried to be resolved [10,13,8]. Until now the motion planning systems developed for the modular robotic systems make specific assumptions based upon their hardware constraints to reduce search space for planning, hence their algorithms might not be valid for other types of modular robotic systems.

* e-mail: k.miyashita@aist.go.jp
And most of the planning systems make plans off-line before execution and do not assume the existence of known/unknown obstacles. When the modular robots work in unknown environments, planning and execution must be interleaved for gathering information through sensing, thus planning should be done in real-time.

Online algorithms for controlling motions of a robot in unknown terrains have been developed by researchers in the fields of theoretical robotics and computer science [1,11]. But they have strong limitations on types and locations of a robot and obstacles in the environments for eliminating the needs to consider several constraints arising from interactions among the robot and the environment. Therefore, their methods cannot be readily applied to motion planning for the modular robots that have complicated physical constraints on possible movements. In this paper, we propose a phased search architecture, RPM (Real-time Planner for Modular Robots), for navigating modular robots in unknown terrains based upon the real-time heuristic search algorithm. In RPM, local path planning for each module and global decision making for the entire system movement are systematically integrated to prevent producing redundant moves of the modules.

In the following of this paper, first, our 2D model of modular robots navigation problem is described and the Real-Time-A* algorithm (RTA*) [7] is introduced to be applied to the problem. Then, we explain a heuristic depression which causes the original RTA* algorithm to produce a superfluous lengthy plan in the problem. We propose a phased search architecture for making a better heuristic estimate of each module with local search and deciding an appropriate movement of the modular robots with global search. And experimental results are shown to validate effectiveness of the proposed architecture for navigating modular robots. And, finally conclusions and future research directions are discussed.

2 Modular Robots Navigation Problem

In this research, we concentrate ourselves into developing computational architecture for real-time navigation of modular robots in unknown environments, where the modular robots are required to reach the goals avoiding the obstacles. In this paper, we assume that the modular robots can move around in the environment but the obstacles and the goals never change their positions. And the modular robots are informed of their initial positions and the location of goals but not about obstacles. The module recognizes existence of obstacles in its neighborhood using its sensing capability and shares the information with the other modules. Hence, the modular robots gradually build an accurate map of the environment as they move around.

To be free from several idiosyncratic hardware constraints and acquire generally applicable findings, we conducted our research using a simple model
of the virtual 2D modular robotic system and built our algorithm on the foundation of a standard heuristic search method.

2.1 2D Modular Robots Model

![Diagram of a 2D modular robot model]

Fig. 1. 2D modular robots simulator

We developed a model of the 2D modular robotic system and implemented a simulator (Fig. 1) in JAVA. In Fig. 1, dark gray blocks in the left represent modular robots, black blocks in the center are obstacles, and light gray blocks below the obstacles show goals.

![Diagram showing module sensing capability and movement]

Fig. 2. Module’s sensing capability, and two types of module’s movement: (1) 90 degree rotation and (2) 180 degree rotation

The movement of the module is constrained by the existence of other modules and obstacles. As essential constraints of general modular robots’ movements, we presume the followings:

1. A module moves relative to another module, called as a *pivot* module.
2. A module must be attached to the pivot module after the movement.
3. The whole modules need to be connected with each other after the motion.
4. While moving, a module must not collide with other modules or obstacles.

We do not assume a module that does not move, which is often called as a fixed base module in the several research of the modular robots. In this paper, the module robots can move to whatever destination far away from their initial location, as long as there is a possible path to the destination. And a module is capable of rotational/rolling motions, but sliding motions are not allowed because they may have large kinetic friction.

To make autonomous movements in unknown environments, each module is capable of sensing its surroundings. The left figure in Fig. 2 shows the area which can be sensed by the module sitting in the center of the figure. The central figure depicts the counter-clockwise rotational motion of the module which makes the module at position 5 as a pivot module. As is shown in the figure, the module makes a rotational movement on the vertex of the pivot module. In this motion, the positions 0, 1, and 3 must not be occupied by other modules or obstacles. The right figure in Fig. 2 puts another example of a module’s movement. To make this motion feasible, the positions 0, 1, 2, 4, e.0 and e.1 must be empty of other modules and obstacles. As explained in Sec. 3.1, these tight physical constraints make the modular robots’ path planning problem intractably hard for a brute-force search algorithm.

2.2 Real-Time Search Algorithm

The problem of motion planning or path planning in robotics can be solved as a state-space search problem. The goal of the state-space search problem is to find a path from an initial state to a goal state. State-space search algorithms can be divided into two categories: one is off-line and another is real-time. When the environment is completely known before the robot begins its traverse and never changes afterwards, some off-line state-space search algorithms, such as the A* algorithm, can find an optimal solution for the problem. But when the robot has partial or no information about the environment before it begins its traverse, the off-line algorithms usually fail to find a solution, not to mention that they can’t find an “optimal” path.

Real-time algorithms, such as RTA* [7], perform sufficient computation to determine a plausible next move making the best of currently available information about the environments, execute that move, then perform further computation to determine the following move, and so on, until the goal state is reached. Hence, the real-time search algorithms allow the robot with sensors to accumulate information of the environments during its movement and exploit the accumulated information to make a plausible plan. These algorithms cannot guarantee to find the optimal solution, but usually find an acceptable solution in a reasonable amount of time. Thus, we apply RTA*
for navigating the modular robots in the unknown environments. The outline of the algorithm is as follows\footnote{The shown algorithm is for the case where the depth of the look-ahead horizon is 1.}:

**Step 1:** Calculate $f(x') = h(x') + k(x, x')$ for each neighbor $x'$ of the current state $x$, where $f(x')$ is the estimated distance from $x$ to the goal via $x'$, $h(x')$ is the current heuristic estimate of the distance from $x'$ to the goal state, and $k(x, x')$ is the distance between $x$ and $x'$.

**Step 2:** Move to a neighbor with the minimum $f(x')$ value. Ties are broken randomly.

**Step 3:** Update the value of $h(x)$ to the second-best of $f(x')$ value.

The algorithm can backtrack to a previously visited state when the estimate of solving the problem from that state plus the cost of returning to the state is less than the estimate cost of going forward from the current cost. By updating $h(x)$ in step 3, the algorithm prevents infinite loops while permitting backtracking when it appears favorable, resulting in a form of single-trial learning through exploration of the problem space. RTA* is guaranteed to be complete in the sense that it will eventually reach the goal under the following conditions: (1) the problem space is finite, (2) all the edge costs are positive, (3) there exists a path from every state to the goal state, and (4) the value of heuristic estimates are finite.

## 3 Phased Search Architecture

We apply RTA* to navigation of the modular robots in the unknown environments. In this paper, we assume that search is done in a centralized fashion by the modular robotic system and only one module can make a movement at each execution. The application of RTA* to navigation of the modular robots is straightforward. Each search state $x$ is a configuration of the modular robot in a specific location. And the heuristic estimate $h(x)$ of the distance between the modular robots and the goal can be a sum of the Diagonal Distance\footnote{The Diagonal Distance is the maximum of the X-axis distance and the Y-axis distance between the state and the goal.} between each module and its closest unoccupied goal location. This is based upon the conjecture that conjunctive goals of modules are completely independent.

### 3.1 Heuristic Depressions

However, except for the very simplistic problems, this naive formulation results in lengthy plans, which are not desirable in the robotic applications because, in general, execution cost of a plan is much higher than planning cost in robotic applications.
Fig. 3. Modular robots trapped in heuristic depressions and schematic view of search process escaping from heuristic depressions

The constraints on module’s motions described in Sec. 2.1 make the modular robots trapped with the obstacles as shown in the left figure of Fig. 3. In the figure, the left-most module attached to the obstacle can leave the obstacle by moving into a direction of either 2, 4, or 7 in Fig. 2. For moving into a direction 2 or 7, a pivot module is required at a location 1 or 6, which are attached to the obstacle. To move into a direction 4, a pivot module is needed at a location 2 or 7. Since every module needs to be connected with each other, the trapped module also needs to be connected with a module either at a location 2 or 7. But since locations behind the module (i.e., locations 0, 3 and 5) are obstacles in Fig 3, for the module to be connected with a module at a location 2 or 7, there must be a module at a location 1 or 6, either of which is attached to the obstacle. This means that modules attached to the obstacles cannot leave them freely. For the modular robots to leave the obstacles, the entire modular robots must go downwards in the figure. Since this is the opposite direction to the goals, the modular robots waste huge steps of motions wondering in the proximity of the obstacles trying to leave them. Therefore, once a motion makes the modular robots attached to the obstacles as shown in Fig. 3, a resultant plan becomes very lengthy. Difficulty of avoiding being trapped in local minima is not unique to this problem, but in general, inaccuracy of heuristic estimates, called a heuristic depression [5], causes a search based planner to make an inefficient plan for filling gaps between a heuristic estimate and a correct distance of the states during search as schematically shown in the right figure of Fig. 3.

3.2 Refining Heuristic Estimate

In Step 1 of the algorithm described in Sec. 2.2, a heuristic estimate of the distance from the current state to the goal state is calculated as a sum of the Diagonal Distance of each module to its closest goal location. The Diagonal Distance metric is excessively underestimated when there are obstacles blocking module’s movements in the environment. With more accurate heuristics, efficiency of the search is expected to improve drastically avoid-
ing being trapped in heuristic depressions. Thus, as a remedy of the original RTA* algorithm, the better estimate for each module in the state is obtained by searching the shortest path to the goal when new obstacles are discovered after a motion of the modular robots. We call this search as Module Path Planning.

To estimate a module’s distance to the goal, we assume that a single module can move by itself (i.e., it can move without help from a pivot module and it does not have to consider about connectivity and collision with the other modules) from its current position to the designated goal. Without the obstacles, the Diagonal Distance gives an accurate distance estimate for the module, which is the shortest path length from the module to the goal. But when there are obstacles in the environment, it might be impossible for the module to reach the goal in the shortest path. In general, a module needs to detour the obstacles along the way to the goal, thus the actual distance from the module to the goal is larger than the Diagonal Distance.

For calculating a more accurate distance than the Diagonal Distance, in Module Path Planning, A* is applied to search for the shortest path from the current position of a module to the unoccupied closest goal when new obstacles are discovered in the environments. And, if a newly estimated value is larger than the original value, the heuristic estimate value of the state is replaced by the new value. Since A* is the best-first heuristic search, it takes exponential time to run in practice. Then, in Module Path Planning, the depth of search can be limited to a certain threshold value to guarantee real-time execution of the algorithm. To avoid being trapped in heuristic depressions described in Sec. 3.1, the special attentions need to be paid to certain topological relationship among goals, obstacles and modules in searching for a path. As explained before, since the module robots tend to be trapped in the proximity of the obstacles, the following heuristics are developed: (1) when a module is attached to a line of obstacles, it should move to the nearby goal that is attached the obstacles, or without such a goal it should go to the closer edge of the obstacles, and (2) when a goal is attached to a line of obstacles, a module should move to the goal.

To be noted is that in Module Path Planning each module is assumed to move independently of the other modules, which keeps the computational complexity of search linear to the number of the modules. Moreover, since Module Path Planning can be executed independently by each module without communicating each other, modules with sufficient computational capability can perform the search fully in parallel. Hence, we propose a phased search architecture (see Fig. 4) for navigating modular robots. In the architecture, when new information of the environment is obtained after a movement, a heuristic estimate of the state is updated in parallel by each module. And based upon the updated estimate, a next move is determined in a centralized computation after checking connectivity of the modules.
4 Experiments

We incorporated the above described ideas in the original RTA* algorithm and developed the real-time path planner for the modular robots, called RPM. In order to evaluate effectiveness of RPM experimentally, we made some preliminary experiments of navigating 2D modular robots in the unknown environments.

The problems we used for experiments are shown in Fig. 5. The modular robots in the left of the figures are requested to move to the goals in the right avoiding the L-shaped obstacles in the center. In an initial state, the modular robots have location information about themselves and the goals but they are ignorant about existence of the obstacles, thus they are put in the unknown environment.

4.1 Results and Analysis

We compare the results of the original RTA* algorithm and RPM (with no limitation on the search depth of Modular Path Planning). In the experiments there are some possibilities that some states are in ties and in such cases the
Table 1. Experiment results of Exp1 and Exp2

<table>
<thead>
<tr>
<th></th>
<th>Exp1</th>
<th>Exp2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RTA</td>
<td>RPM</td>
</tr>
<tr>
<td>0 - 99</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100 - 199</td>
<td>40</td>
<td>95</td>
</tr>
<tr>
<td>200 - 299</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>300 - 399</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>400 - 499</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>500 - 599</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600 - 699</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>700 - 799</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>800 - 899</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900 - 999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1000 -</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>Sum</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

ties are broken randomly. Therefore we repeated solving the same problem 100 times with the different random seed values in the experiments.

The results of the experiments are summarized in Table 1. In the tables, RTA* means the original Real-Time A* algorithm and RPM indicates the RPM algorithm that combines RTA* with estimate refinement by Module Path Planning. Since, in the experiments of Exp2, the original Real-Time A* algorithm failed to find a solution within limitation of available memory size (1G byte), only the results of RPM are shown in the tables.

The experiments are evaluated using the length of resultant plans. The first row of the left table in Table 1 shows the average of plan length in 100 experiments. The second row presents the standard deviation of plan length. The third and fourth rows are the minimum and maximum plan length obtained in 100 experiments respectively. As shown in the table, RPM outperforms the original RTA* algorithm considerably in Exp1. The right table in Table 1 shows the distribution of plan length in 100 experiments. Each row in the table shows the number of plans whose length is within the designated range. The table presents that in Exp1 RPM succeeded to find good solutions (less than 200 moves) in most of the cases of 100 random trials. On the contrary, RTA* produces redundant plans (over 1000 moves) in 60 cases. This shows that in RTA* the modular robots are caught in heuristic depressions and forced to wander around for getting them over.

From the table, it is also clear that Exp2 is more difficult than Exp1 for modular robots. This is because in Exp2 some goals are attached to the obstacles and modules must approach those goals along the obstacles due to physical constraints explained in Sec. 2.1, thus versatility of motion selection for modules are more restricted than in Exp1. In such a difficult problem, RPM still succeeded to find a good solution for many cases.
5 Conclusions

In this paper, it is experimentally shown that RPM is the effective algorithm for navigating the modular robots in the unknown environments. RPM extends RTA* by refining heuristic estimate of the states by executing local search upon discovery of new obstacles, thus overcoming difficulty of heuristic depressions. In the future study, we are planning to develop a distributed version of RPM that also allows simultaneous motion executions by multiple modules. Then, RPM will be embedded in the 3D hardware modular robots.

References