Distributed, Dynamic, and Autonomous Reconfiguration Planning for Chain-Type Self-Reconfigurable Robots

Feili Hou, Wei-Min Shen

Presented by: Ayan Dutta
Key Features

• Reconfiguration planning.
• Dynamic and distributed.
• Chain type self-reconfigurable robots.
• Unlabeled complex configurations.
• Reconfiguration is done only in modules which are part of ‘to-be-reconfigured’ sub-graph.
Reconfiguration planning problem

• Nodes do not have unique IDs.
• Modules can communicate with only immediate neighbors.
• Two configurations are adjacent if one can be transformed into the other by one set of reconfigure actions, which is an attach action followed by a detach action, together with some motion of chains.
SuperBot and Reconfiguration process

Fig. 1. A SuperBot configuration and its modular graph

Fig. 2. A set of reconfiguration actions to transform from one configuration to its adjacent configuration
Distributed configuration comparison

- Sub-graphs which needs to be reconfigured and which does not need to be: TBR and NTBR.

- Reconfiguration happens only in TBR sub-graph.
- Each module can decide whether it belongs to the TBR subgraph by communicating with their immediate neighbors.
Goal configuration representation

• Graph of the goal configuration is informed to every module in the robot.
• New way to represent the unlabeled graph called configuration string is proposed here.
• Each module has a CN variable: where CN[i] denotes the number of modules connected to its i-th connector.

$$\text{Sum}_\text{Modules} = \sum_{i=1}^{NC} CN[i] + 1$$

• A node is the center if all of its CN elements are less than half of Sum_Modules.
• Starting from the center node, we traverse the modular graph of the goal configuration by Depth-First-Search (DFS).
Configuration String of Goal (CSG)

<table>
<thead>
<tr>
<th>DFS Order</th>
<th>Node ID</th>
<th>CN Value</th>
<th>DFS Order</th>
<th>Node ID</th>
<th>CN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>[10 6 4 0]</td>
<td>12</td>
<td>L</td>
<td>[15 5 0 0]</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>[11 3 3 3]</td>
<td>13</td>
<td>M</td>
<td>[16 2 1 1]</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>[18 1 1 0]</td>
<td>14</td>
<td>N</td>
<td>[19 1 0 0]</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>[20 0 0 0]</td>
<td>15</td>
<td>O</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>[20 0 0 0]</td>
<td>16</td>
<td>P</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>[18 1 1 0]</td>
<td>17</td>
<td>Q</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>7</td>
<td>G</td>
<td>[20 0 0 0]</td>
<td>18</td>
<td>R</td>
<td>[17 1 1 1]</td>
</tr>
<tr>
<td>8</td>
<td>H</td>
<td>[20 0 0 0]</td>
<td>19</td>
<td>S</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>9</td>
<td>I</td>
<td>[18 1 1 0]</td>
<td>20</td>
<td>T</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>10</td>
<td>J</td>
<td>[20 0 0 0]</td>
<td>21</td>
<td>U</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>11</td>
<td>K</td>
<td>[20 0 0 0]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
CSG = [10 \ 6 \ 4 \ 0][11 \ 3 \ 3 \ 3][18 \ 1 \ 1 \ 0][20 \ 0 \ 0 \ 0][20 \ 0 \ 0 \ 0][18 \ 1 \ 1 \ 0][20 \ 0 \ 0 \ 0][20 \ 0 \ 0 \ 0][15 \ 5 \ 0 \ 0][16 \ 2 \ 1 \ 1][19 \ 1 \ 0 \ 0][20 \ 0 \ 0 \ 0][20 \ 0 \ 0 \ 0][17 \ 1 \ 1 \ 1][20 \ 0 \ 0 \ 0][20 \ 0 \ 0 \ 0][20 \ 0 \ 0 \ 0][20 \ 0 \ 0 \ 0] \]
Current Configuration Recognition

• Done by exploring own CN value, and then collectively performing the configuration comparison.

• **Constraint**: the robot keeps connected throughout the reconfiguration process.

• First, modules decide Sum_Modules.

• *Probe* message: received from a connector and describes the number of modules connected through it.

\[
CN[k] = \text{Sum} \_\text{Modules} - 1 - \sum_{i \neq k} CN[i] \tag{4}
\]

, and send out a *probe* message through connector k with value

\[
probe = \sum_{i \neq k} CN[i] + 1 \tag{5}
\]
TABLE II  CN VALUES OF THE MODULES IN FIG 3(A)

<table>
<thead>
<tr>
<th>Node ID</th>
<th>CN Value</th>
<th>Node ID</th>
<th>CN Value</th>
<th>Node ID</th>
<th>CN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[10 6 4 0]</td>
<td>8</td>
<td>[17 2 1 0]</td>
<td>15</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>2</td>
<td>[11 5 4 0]</td>
<td>9</td>
<td>[19 1 0 0]</td>
<td>16</td>
<td>[15 5 0 0]</td>
</tr>
<tr>
<td>3</td>
<td>[16 3 1 0]</td>
<td>10</td>
<td>[20 0 0 0]</td>
<td>17</td>
<td>[16 2 1 1]</td>
</tr>
<tr>
<td>4</td>
<td>[18 1 1 0]</td>
<td>11</td>
<td>[20 0 0 0]</td>
<td>18</td>
<td>[19 1 0 0]</td>
</tr>
<tr>
<td>5</td>
<td>[20 0 0 0]</td>
<td>12</td>
<td>[17 1 1 1]</td>
<td>19</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>6</td>
<td>[20 0 0 0]</td>
<td>13</td>
<td>[20 0 0 0]</td>
<td>20</td>
<td>[20 0 0 0]</td>
</tr>
<tr>
<td>7</td>
<td>[20 0 0 0]</td>
<td>14</td>
<td>[20 0 0 0]</td>
<td>21</td>
<td>[20 0 0 0]</td>
</tr>
</tbody>
</table>
Configuration comparison

• Each node will do the configuration comparison locally by comparing its CN value with the corresponding goal CN in the CSG, and decide whether it belongs to the TBR sub-graph.
• A node can decide whether it is the center node by checking its CN.
• An index message is initiated by the center node and propagated to trigger the local configuration comparison of other modules.
• The index message sent to a node describes the location of its goal CN in the CSG string.

\[
index[i] = k + \sum_{j=1}^{i-1} CN[j] + 1
\]
Reconfiguration process

Fig. 4 reconfiguration steps of the TBR sub-graph in Fig. 3(a) to a line
Discussion of the algorithm

- **Dynamic and on-line** planning.
- Modules are homogeneous that do not have IDs.
- The robot can self-reconfigure **without any prior knowledge** of preplanned **reconfiguration steps**.
- **Effort efficient**: only TBR is reconfigured, NTBR is extracted out using comparison.
- The communication complexity of the comparison is \( O(N) \).
- It is **scalable** and time **efficient**.
- Local information and local communication with neighbors.
- It is **fault tolerant**: robust to message loss to some extent.
Thank you...

Presented by: Ayan Dutta
University of Nebraska at Omaha, USA.