Developmental Co-evolution for Roombots

Master Project Midterm Presentation

Ebru Aydın

Advisors:
Prof. Auke Jan Ijspeert
Rico Möckel
Jesse van den Kieboom
Alexander Spröwitz
Project Description:

Developmental Co-evolution for Roombots

Developmental Encoding:

- Grammatical Approach
- L-systems
  ex: $A \rightarrow A-B-A$, $B \rightarrow B+A+B$

Co-evolution:

- Evolution of configuration and controller
  - Roombots
  - CPG, coupled phase oscillators
Previous work: Demo Lab

Projects from Demo Lab: Hornby, Pollack, Lipson

- Generative Representations for Design Automation
  Static structures, 2D and 3D robots are evolved.

- The Golem Project
Previous work: Demo Lab

Generative encoding builds creatures with better locomotion ability than non-generative encoding.

Generative encoding brings:
- Scalability through self-similar structures
- Hierarchical structure
- Compact representation
Co-evolution for "Adam", D. Marbach

- Modular robot (Yamor style)
- Evolution of configuration and controller
- Harmonic oscillators for controller
- Tree-based encoding
- Evolved creatures are fitter than hand-designed ones
Co-evolution for Roombots:

• Configuration is evolved as in Marbach’s project, not the morphology
• Building blocks : Roombots
• Controller :
  Coupled phase oscillators, 3 drive function
• Each module : 3 DOF
• Adding a module (15 parameter):
  • Connection
    type, face of each module, phase lag (4)
  • For each servo : amplitude, offset, drive function (9)
  • Between servos in a module : Phase lag (2)
## Timeline

<table>
<thead>
<tr>
<th>Number</th>
<th>Task</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Literature Review</td>
<td>2/22/2010</td>
<td>4/11/2010</td>
<td>77</td>
</tr>
<tr>
<td>3</td>
<td>Phenotype to genotype conversion</td>
<td>3/15/2010</td>
<td>4/5/2010</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Evolutionary Part Design and implementation</td>
<td>3/7/2010</td>
<td>4/21/2010</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Dispatcher implementation</td>
<td>3/27/2010</td>
<td>4/16/2010</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>Controller implementation</td>
<td>3/31/2010</td>
<td>4/19/2010</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>The system will be ready for experiments</td>
<td>4/26/2010</td>
<td>4/26/2010</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Experiments and result analysis</td>
<td>4/26/2010</td>
<td>5/6/2010</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>Passive elements design and implementation</td>
<td>5/6/2010</td>
<td>5/26/2010</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>Additional experiment tasks</td>
<td>5/17/2010</td>
<td>6/7/2010</td>
<td>15</td>
</tr>
</tbody>
</table>
L-system

**Terminal Rules (Build Commands):**

- `{ }` Block operator
- `[ ]` Save and store the state
- Change face
- Change oscillator type
- Change phase lag
- Change amplitude
- Change offset
- Add module

**State:** Module, face, phase lag
Sample Grammar:

\[ r1 \rightarrow S_{\text{CPL}} \ 3, \text{ADD C3X PAR}, \text{COT S1 LOCKED} \]
\[ r2 \rightarrow 2\{ S_{\text{CF}} \ 2, \ [ \ r1 \ ] \} \]
\[ r3 \rightarrow S_{\text{CF}} \ 7, \ [ \ r1 \ ], \ r2 \]
\[ r4 \rightarrow \text{COT S1 LOCKED}, \ [ \ r3 \ ] \]

Rewritten string:

\text{COT S1 LOCKED, [ S_{\text{CF}} \ 7, [ S_{\text{CPL}} \ 3, \text{ADD C3X PAR}, \text{COT S1 LOCKED }], 2\{ S_{\text{CF}} \ 2, [ S_{\text{CPL}} \ 3, \text{ADD C3X PAR}, \text{COT S1 LOCKED } ]\}]}
Sample creature:
Optimizer

- Evolutionary algorithm
- Evolving grammars

a- Mutations:
   1. Change parameter of a build command
   2. Add rule to rewrite rule
   3. Remove rule from rewrite rule

b- Crossover: Use an index, copy the rules from parents:
   \[ r1 \quad r2 \quad r3 \quad r4 \]
Dispatcher

Dispatcher:

- Webots controller
Fitness: Direct distance, measured in 20 seconds
Future work:

- Experiments, fitness (jumping, efficiency)
- Passive elements (spring, hinge joints, passive leg)
- Sensory feedback
- Analysis
- Report
Thank you for your attention!