On Scalability Issues in Reinforcement Learning for Modular Robots

Paulina Varshavskaya, Leslie Pack Kaelbling and Daniela Rus
• motivate reinforcement learning (RL)
• larger modular system challenges
• a specific solution
• a general solution
• experiments in simulation
• related current work
RL for self-reconfigurable robots

1. automated controller design

MTRAN-II controller: genetic algorithms (Kamimura et al 2004)
Molecule controller: genetic algorithms (Mytilinaios et al 2004)
Telecube primitives and controller: genetic programming (Kubica and Rieffel 2002)
RL for self-reconfigurable robots

1. automated controller design

2. online adaptation
   - from the point of view of each robot or module
   - limited knowledge and resources

moving from 1. to 2.
Task: locomotion in modular robots

- lattice-based robots

Molecule Kotay & Rus 2005

simulated generalized lattice-based robot
From each agent’s point of view

state: global configuration of robot

local observation: Moore neighborhood

actions: 8 directions + NOP

reward: Eastward displacement along x axis

assumptions: primitive “physics”, no disconnections, failed actions don’t execute, synchronous execution
Large modular sytem challenges

- partial observability
cannot use Markov-assuming “nice” algorithms

- large observation-action spaces
  \[2^8 \text{ observations without obstacles} \times 9 \text{ actions}\]
  \[3^8 \text{ observations with obstacles} \times 9 \text{ actions}\]
Partial observability

in a multi-agent partially observable Markov Decision Process (POMDP)

direct policy search using gradient ascent in policy space (GAPS)

*Peshkin 2001*

\[
V(\theta) = E_\theta [R] \\
R = \sum_{t=1}^{T} \gamma^t r_t
\]
GAPS

each agent executes a parameterized policy

\[ \pi(\theta) = P(a_t \mid o_t, \theta) = \frac{e^{\beta_t \theta(o_t, a_t)}}{\sum_{a'} e^{\beta_t \theta(o_t, a')}} \]

\( \beta_t \) is “temperature”

at time \( t \):

- observe \( o_t \)
- select \( a_t \) according to policy
- execute \( a_t \)
- receive reward \( r_t \)
- keep an execution trace

at end of episode:

update parameters \( \theta \)
Large modular system challenges

- partial observability

- large observation-action spaces

\[ 2^8 \text{ observations without obstacles} \quad 2,304 \text{ parameters} \]
\[ 3^8 \text{ observations with obstacles} \quad 59,049 \text{ parameters} \]

- acting module
- neighbor present
- empty lattice cell
Specific solution: incremental learning

<table>
<thead>
<tr>
<th>Modules</th>
<th>Observations</th>
<th>Actions</th>
<th>Total Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>x 2</td>
<td>x 2</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>x 2</td>
<td>x 4</td>
<td>162</td>
</tr>
<tr>
<td>4</td>
<td>x 4</td>
<td>x 8</td>
<td>414</td>
</tr>
<tr>
<td>&gt;4</td>
<td>x 8</td>
<td>x 4</td>
<td>&gt;2304</td>
</tr>
</tbody>
</table>

Possible observations x 9 possible actions = total number of parameters

- acting module
- neighbor present
- empty lattice cell
Incremental GAPS (IGAPS) performance

Mean convergence times comparison

Mean average reward comparison

Varshavskaya et al 2004
A specific solution

additive nature of modular robots

unclear applicability to other tasks and systems

faster RL but not fast enough for online adaptation
Desirable general solution

<table>
<thead>
<tr>
<th>possible observations</th>
<th>total number of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 modules</td>
<td>small ( n )</td>
</tr>
<tr>
<td></td>
<td>( \times 2 )</td>
</tr>
<tr>
<td></td>
<td>( \times 2 )</td>
</tr>
<tr>
<td>3 modules</td>
<td>small ( n )</td>
</tr>
<tr>
<td></td>
<td>( \times 2 )</td>
</tr>
<tr>
<td></td>
<td>( \times 4 )</td>
</tr>
<tr>
<td></td>
<td>( \times 8 )</td>
</tr>
<tr>
<td>4 modules</td>
<td>small ( n )</td>
</tr>
<tr>
<td></td>
<td>( \times 4 )</td>
</tr>
<tr>
<td></td>
<td>( \times 4 )</td>
</tr>
<tr>
<td></td>
<td>( \times 8 )</td>
</tr>
<tr>
<td>9+ modules</td>
<td>small ( n )</td>
</tr>
<tr>
<td></td>
<td>( \times 4 )</td>
</tr>
<tr>
<td></td>
<td>( \times 8 )</td>
</tr>
<tr>
<td></td>
<td>( \times 8 )</td>
</tr>
</tbody>
</table>

- acting module
- neighbor present
- empty lattice cell

Paulina Varshavskaya, Leslie Pack Kaelbling, Daniela Rus

19 August 2006
Features

- acting module
- neighbor present
- empty space
- don’t care

\[ \varphi_7 = \begin{cases} 
1 & \text{if upper left corner full } \& \ a = \text{NE} \\
0 & \text{otherwise} 
\end{cases} \]

\[ \varphi_{88} = \begin{cases} 
1 & \text{if upper right corner empty } \& \ a = \text{SE} \\
0 & \text{otherwise} 
\end{cases} \]

\[ \varphi_{45} = \begin{cases} 
1 & \text{if empty row in front } \& \ a = \text{SE} \\
0 & \text{otherwise} 
\end{cases} \]

\[ \varphi_{75} = \begin{cases} 
1 & \text{if upper left corner empty } \& \ a = \text{W} \\
0 & \text{otherwise} 
\end{cases} \]
Approximation by feature spaces

define a number of feature functions over the observation-action space

\[
\Phi(a, o) = \begin{bmatrix}
\varphi_1(a, o) \\
\vdots \\
\varphi_n(a, o)
\end{bmatrix}
\]

policy computed from the dot product of the feature vector and parameters \( \theta \)

\[
P(a_t \mid o_t, \theta) = \frac{e^{\beta_t \Phi(a_t, o_t) \cdot \theta}}{\sum_{a'} e^{\beta_t \Phi(a', o_t) \cdot \theta}}
\]

- a few features approximate the desired space
- learning with a modified log-linear GAPS (LLGAPS)
Full feature set

144 features

- acting module
- neighbor present
- empty space
- don’t care
Performance comparison

Performance as learning progresses:
Smoothed average reward over 10 runs

Mean convergence times comparison
Learned locomotion
Summary so far

- reinforcement learning for modular robots with more realistic observability assumptions
- learning algorithm with feature representation
- results in locomotion by self-reconfiguration
Advantages of a feature representation

• faster learning

• number of parameters independent of robot size or observation space size

• domain knowledge
  – “don’t try to move into an occupied cell”

• features can be designed for many tasks and robots
Current work: asynchronous execution

$2^4$ observations, 11 actions
only 220 features
Current work: large system locomotion

20 modules

70 modules
Current work: complex reward

distribute subtasks within each module
Current work: locomotion in truss robots

MultiShadySim Detweiler et al 2006

x4 Shady Vona et al 2006
Questions?

project sponsored by Boeing Corporation