On Scalability Issues in Reinforcement Learning for Modular Robots

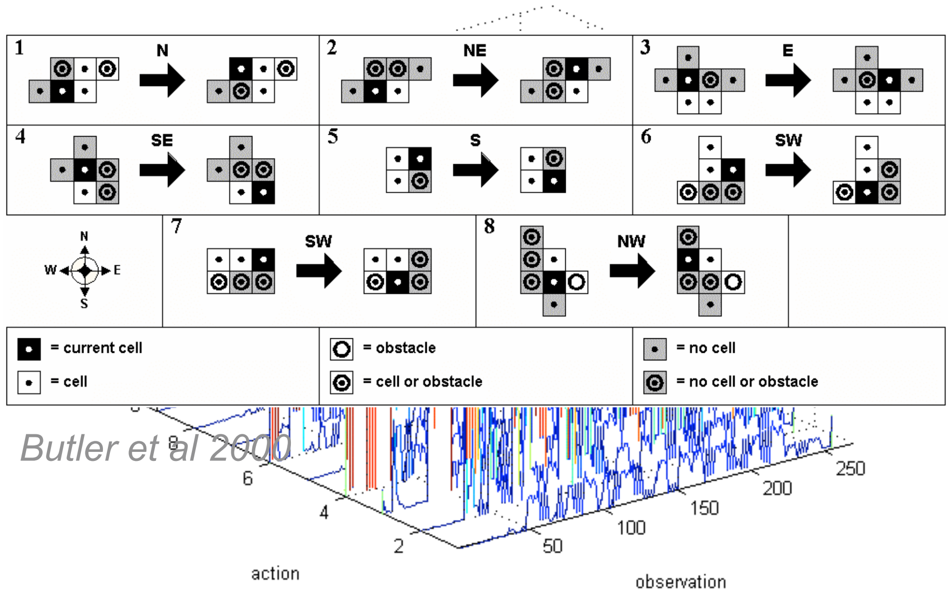
Paulina Varshavskaya, Leslie Pack Kaelbling and Daniela Rus

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

- 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



Butler et al 2000

MTRAN-II controller: genetic algorithms (*Kamimura et al 2004)*

Molecube controller: genetic algorithms (M*ytilinaios et al 2004)*

Telecube primitives and controller: genetic programming (*Kubica and Rieffel 2002)*

RL for self-reconfigurable robots

1. automated controller design

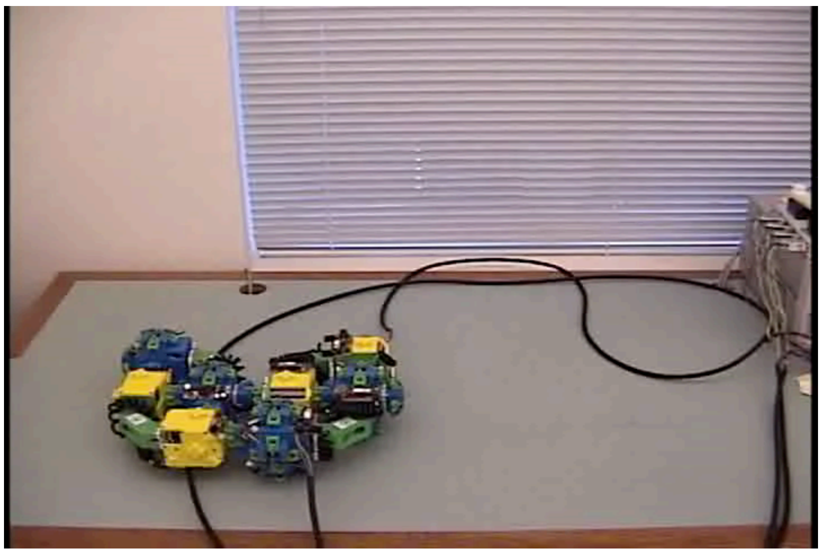
2. online adaptation
   - from the point of view of each robot or module
   - limited knowledge and resources

moving from 1. to 2.

Task: locomotion in modular robots

- lattice-based robots

x4



Molecule *Kotay & Rus 2005*

simulated generalized lattice-based robot

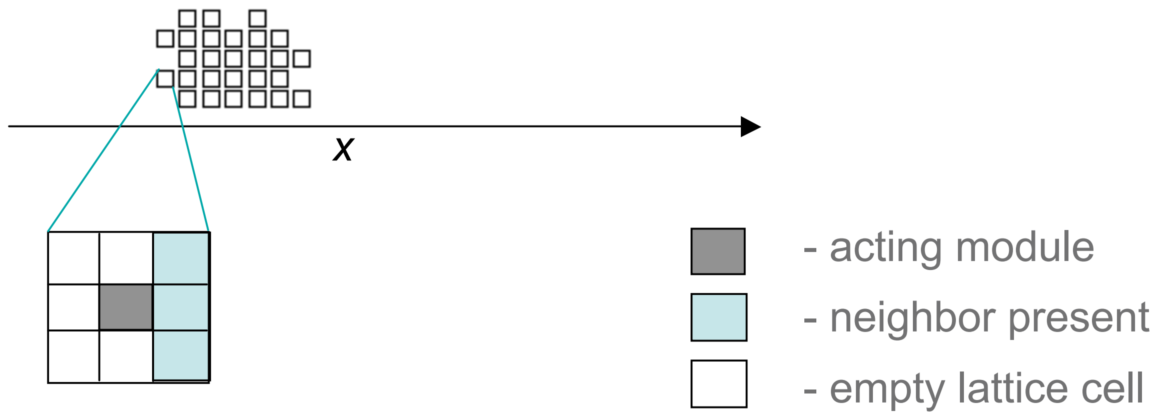
From each agent's point of view

state: global configuration of robot ← unknown

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

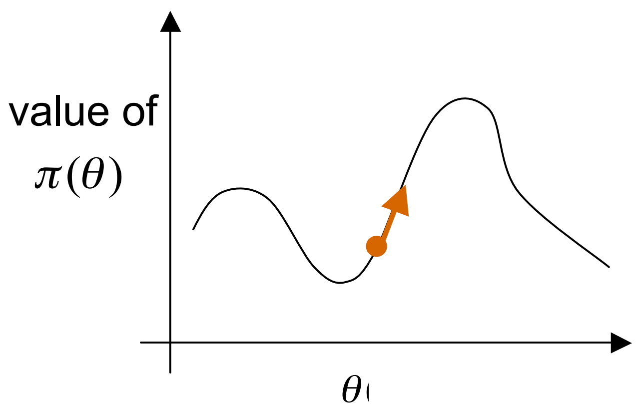
  2^8 observations without obstacles x 9 actions
  3^8 observations with obstacles x 9 actions

Partial observability

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

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

Peshkin 2001



Value V(θ) of policy π(θ)

$$V(\theta) = E_\theta[R]$$

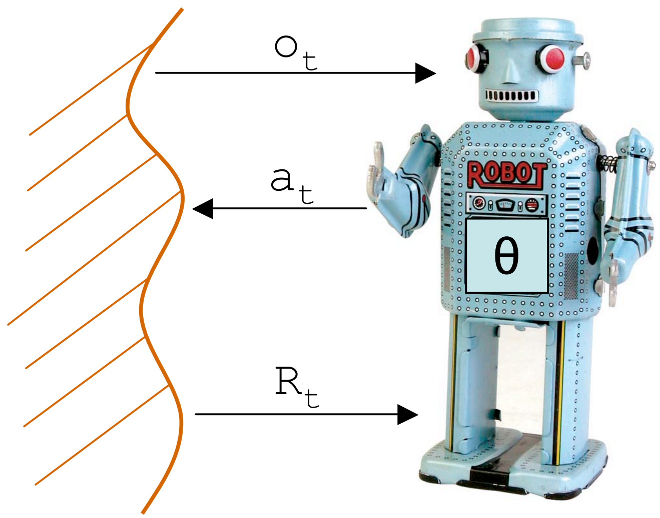
$$R = \sum_{t=1}^{T} \gamma^t r_t$$

GAPS

each agent executes
a parameterized policy

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

β_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 θ
```

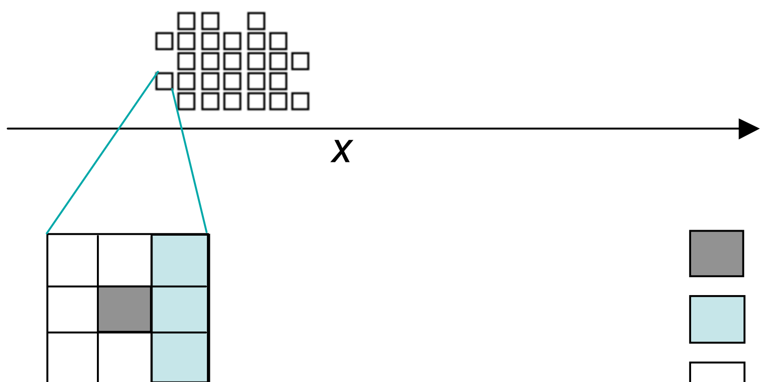
CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Large modular sytem challenges

- partial observability

- large observation-action spaces

2^8 observations without obstacles	2,304 parameters
3^8 observations with obstacles	59,049 parameters

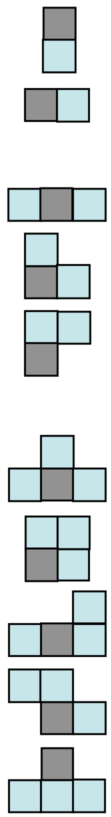


- acting module
- neighbor present
- empty lattice cell

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Specific solution: incremental learning

possible observations x 9 possible actions = total number of parameters

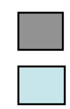


2 modules
x 2
x 2
= 36

3 modules
x 2
x 4
x 8
=162

4 modules
x 4
x 4
x 8
x 8
x 4
= 414

9+ modules
= 2,304

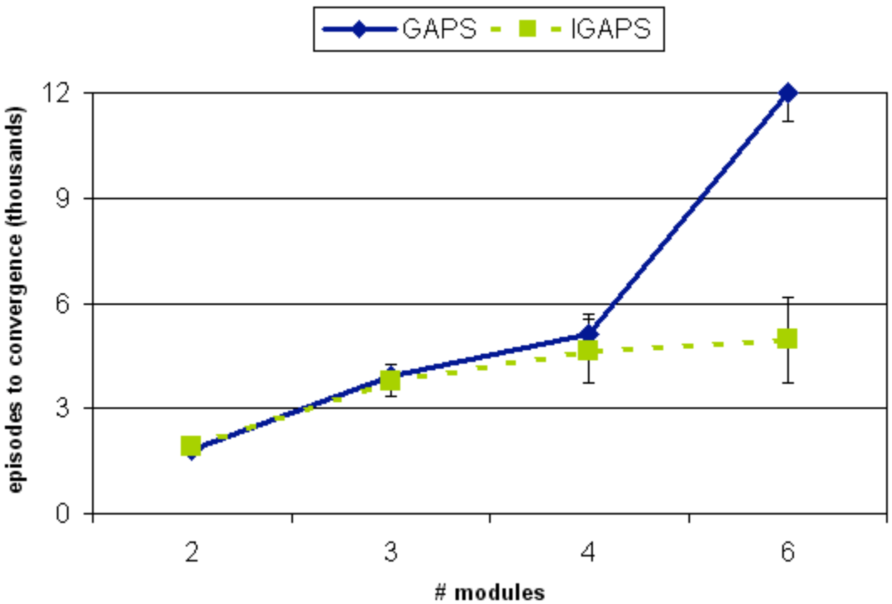


- acting module
- neighbor present
- empty lattice cell

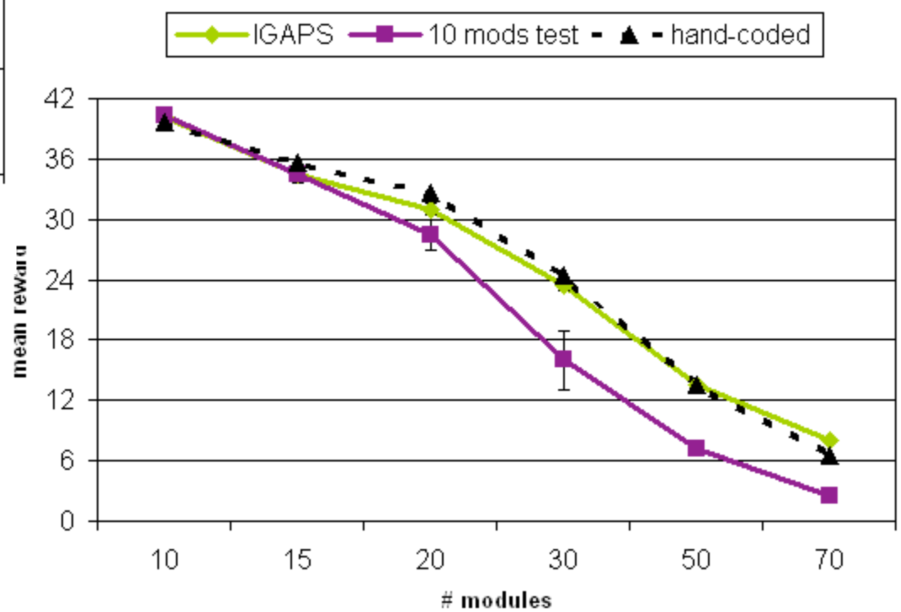
MIT CSAIL COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

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

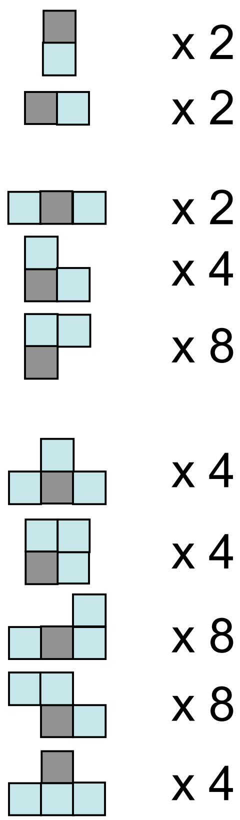
possible observations | total number of parameters

2 modules

3 modules

4 modules

9+ modules

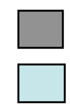


small *n*

small *n*

small *n*

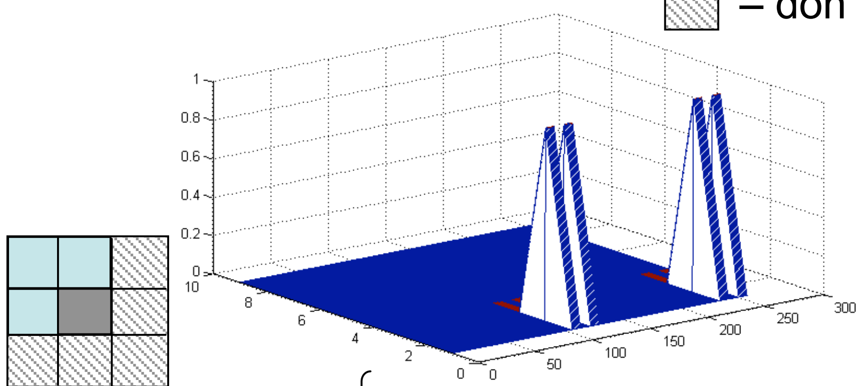
small *n*



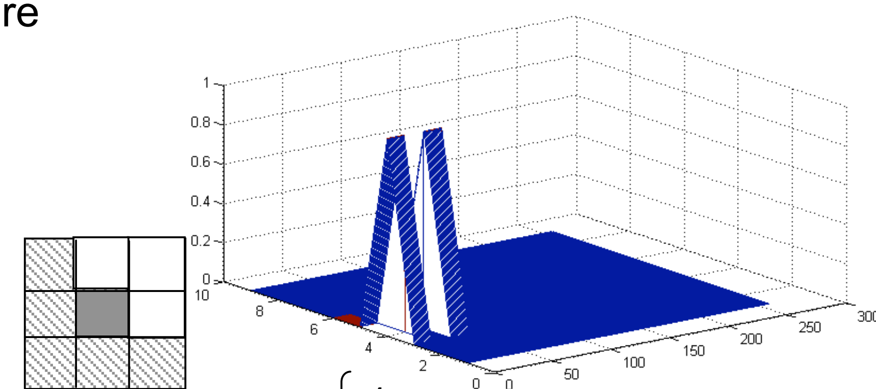
Features

– acting module – neighbor present – empty space

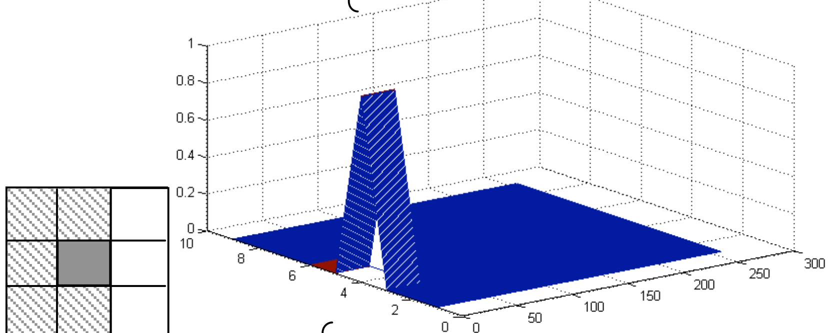
– don't care



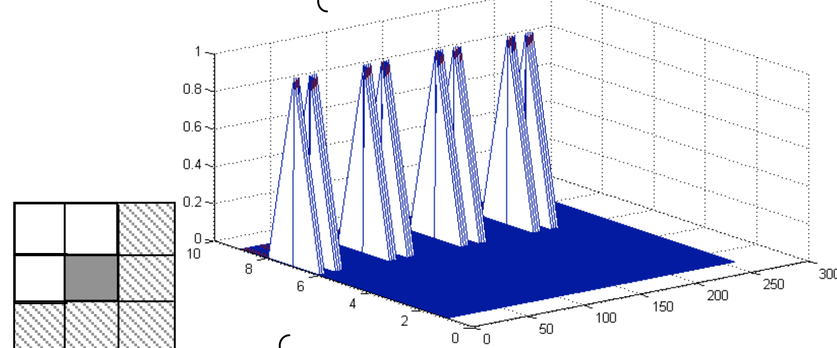
$$\varphi_7 = \begin{cases} 1 & \text{if upper left corner full \& } a=NE \\ 0 & \text{otherwise} \end{cases}$$



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



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



$$\varphi_{75} = \begin{cases} 1 & \text{if upper left corner empty \& } a=W \\ 0 & \text{otherwise} \end{cases}$$

Approximation by feature spaces

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

$$\Phi(a,o) = \begin{vmatrix} \varphi_1(a,o) \\ \cdots \\ \varphi_n(a,o) \end{vmatrix}$$

policy computed from the dot product of the feature vector and parameters θ

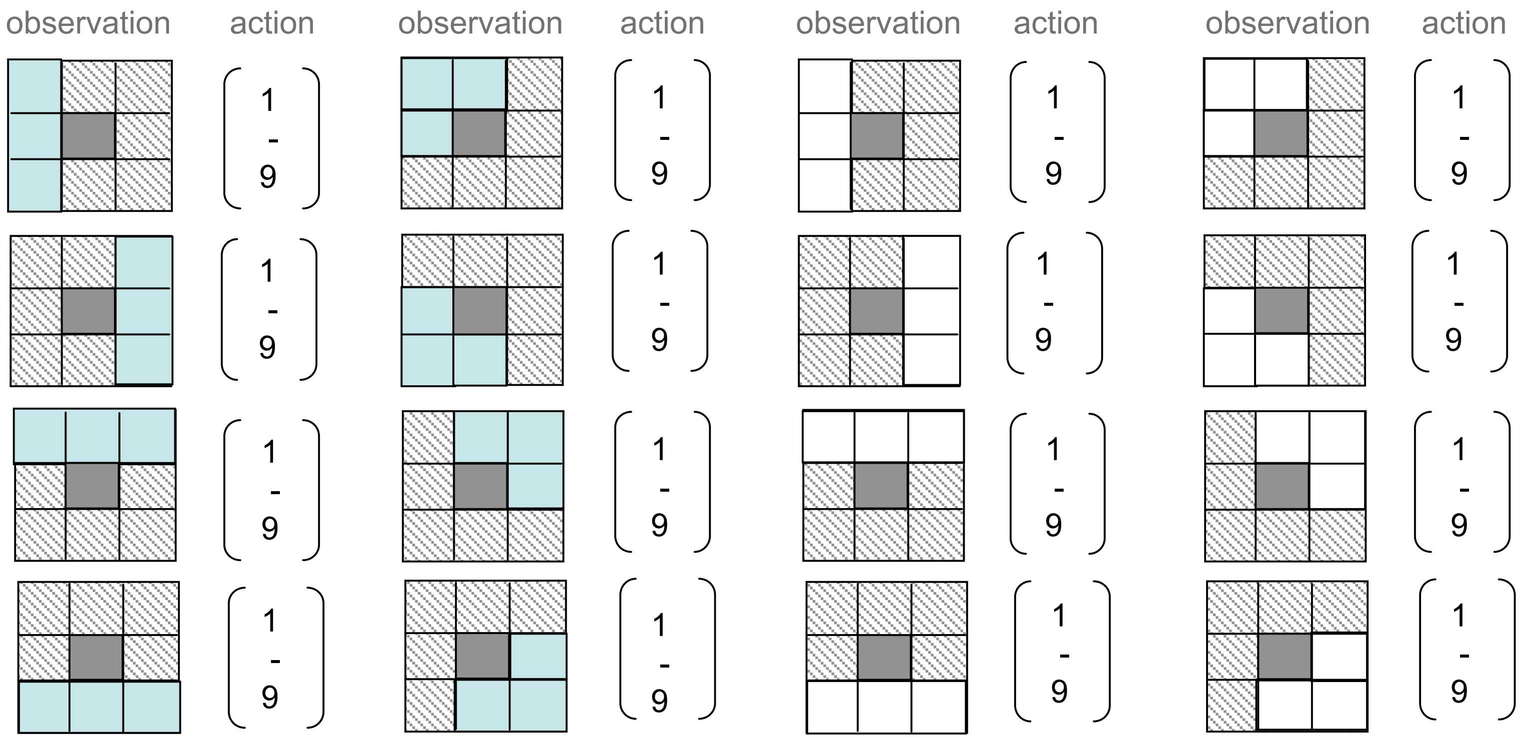
$$P(a_t \mid o_t, \theta) = \frac{e^{\beta_t \Phi(a_t, o_t)\cdot\theta}}{\sum_{a'} e^{\beta_t \Phi(a', o_t)\cdot\theta}}$$

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

CSAIL

MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Full feature set

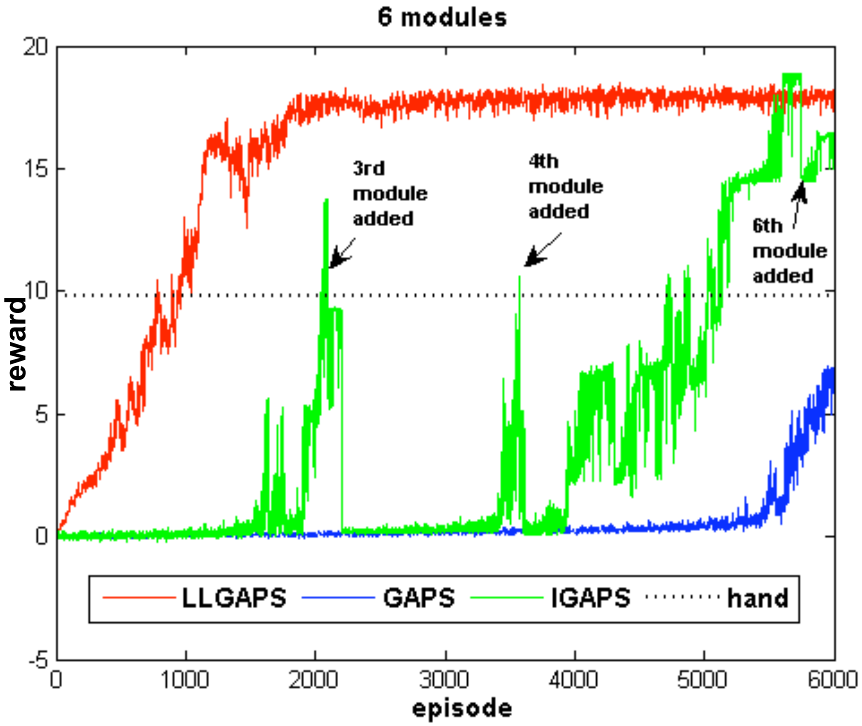


144 features

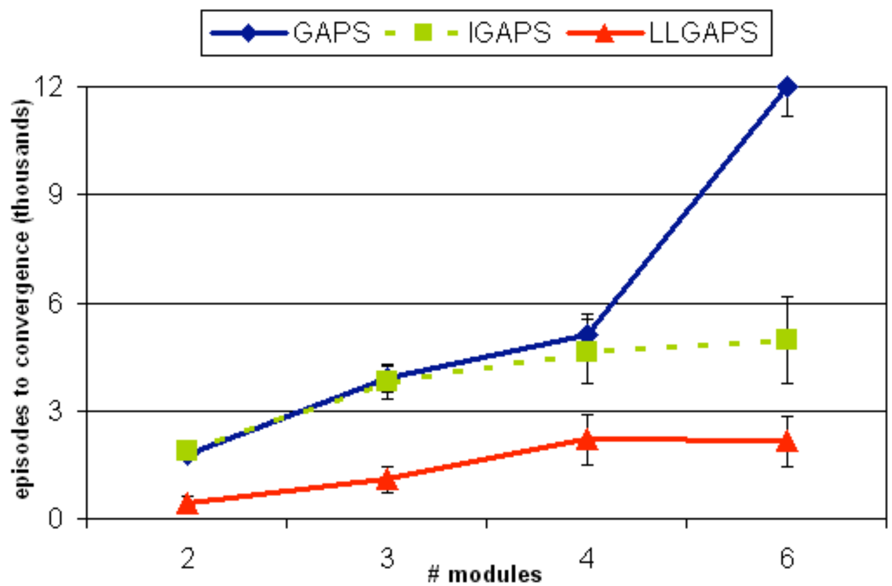
– acting module
– neighbor present
– empty space
– don't care

Performance comparison

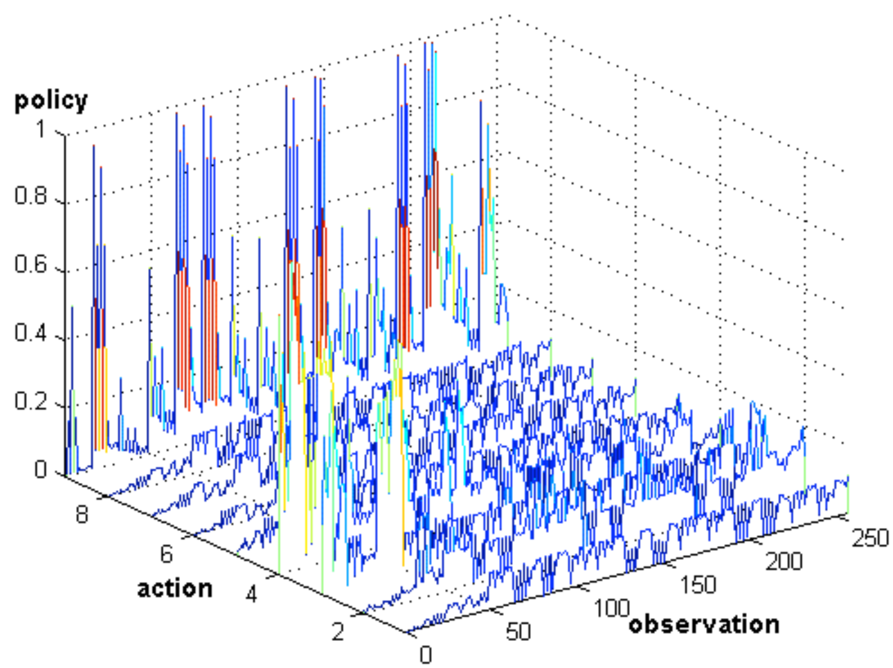
Performance as learning progresses:
Smoothed average reward over 10 runs



Mean convergence times comparison



Learned locomotion



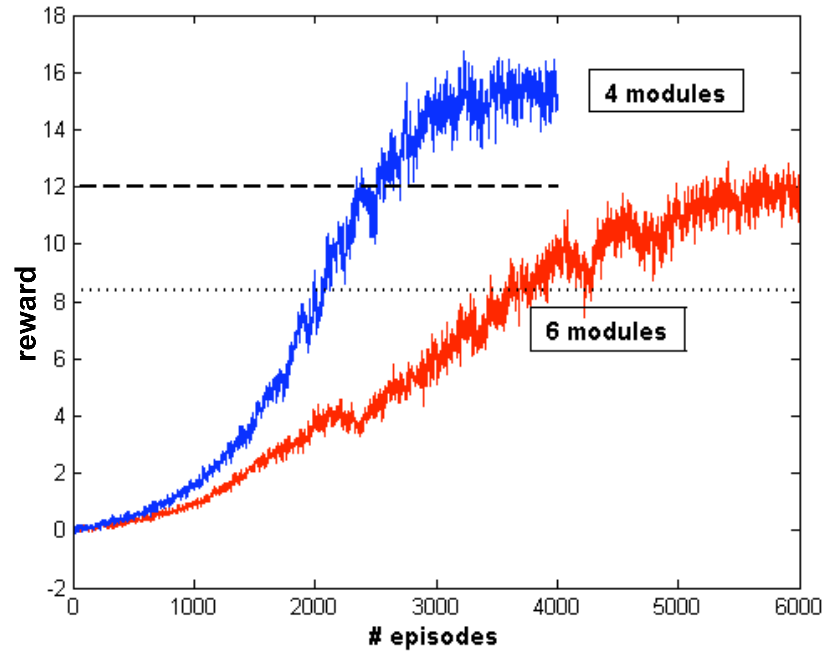
Summary so far

- 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

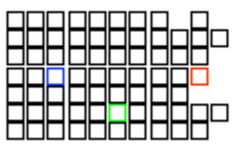
Current work: asynchronous execution



2^4 observations, 11 actions
only 220 features

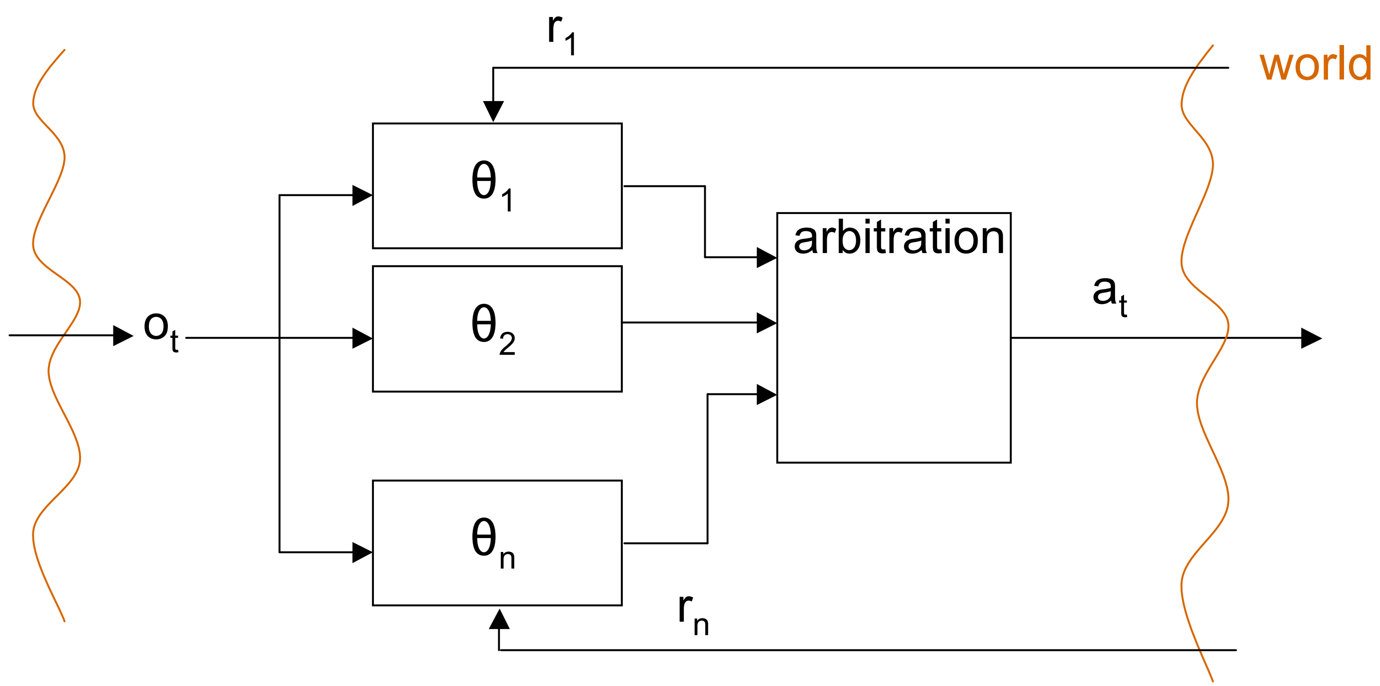
Current work: large system locomotion

20 modules



70 modules

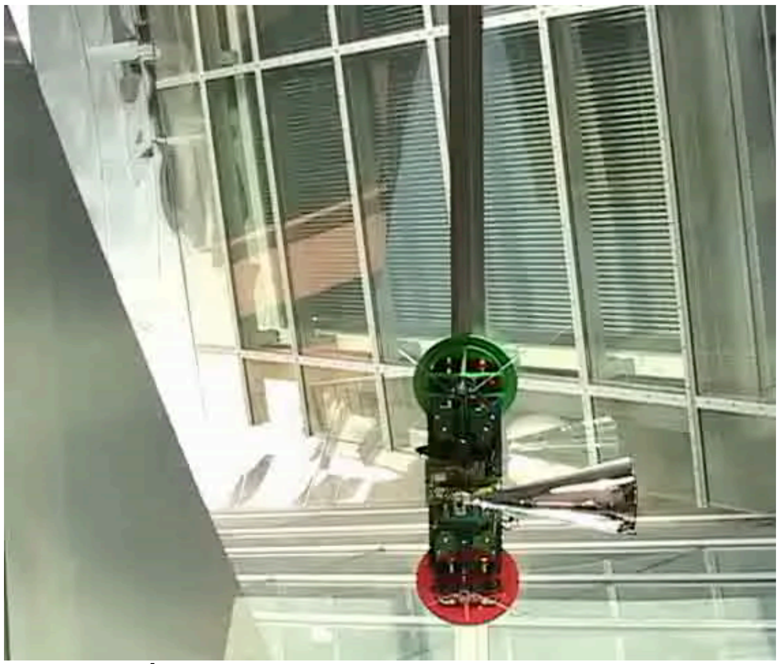
Current work: complex reward



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