Robot Swarms as Ensembles of Cooperating Components

Matthias Hölzl
With contributions from Martin Wirsing, Annabelle Klarl

AWASS
Lucca, June 24, 2013
The Task

- Robots cleaning an exhibition area
marXbot

- Miniature mobile robot developed by EPFL
- Rough-terrain mobility
- Robots can dock to other robots
- Many sensors
  - Proximity sensors
  - Gyroscope
  - 3D accelerometer
  - RFID reader
  - Cameras
  - …
- ARM-based Linux system
- Gripper for picking up items
Swarm Robotics
Problems

- Noise, sensor resolution
- Extracting information from sensor data
- Unforeseen situations
- Uncertainty about the environment
- Performing complex actions when intermediate results are uncertain
- ...

Mateias Hölzl
Action Logics

- Logics that can represent change over time
- Probabilistic behavior can be modeled (but is cumbersome)
Markov Decision Processes

- $s, n, w, e / 0.05 / -0.1$
- $s / 0.9 / -0.1$
- $e, w / 0.025 / -0.1$
- $n / 0.9 / -0.1$
- $e, w / 0.025 / -0.1$
- $s, n / ...$
- $w / ...$
- $e / ...$
- $s, n / ...$

Matthias Hölzl
Markov Decision Processes

Decide Activity

- gotoClub
- watchTV

In Club
- dance p = 0.5

Dancing Alone

Dancing With Partner
- dance p = 0.5
- flirt

Drinking
- drinkBeer

Watching TV

Oh, oh
- p = 0.05

Oh, no
- p = 0.95
MDPs: Strategies

State | TV       | CDB       | CDF       |
------|----------|-----------|-----------|
DA    | watchTV  | goToClub  | goToClub  |
IC    | drinkBeer| drinkBeer | dance     |
DWP   | flirt    | flirt     | flirt     |
Utility | 0.1      | −0.05(*)  | −1.975(*) |

(*) 0.05 + (−0.1)

(**) 0.05 + (0.5 × 0.2) + 0.5 × (0.25 + (0.05 × 5) + (0.95 × −5))
Reinforcement Learning

General idea:

- Figure out the *expected value* of each *action* in each state
- Pick the action with the highest expected value (most of the time)
- Update the expectations according to the actual rewards
How well does this work?

- Rather well for small problems
- But: state explosion
Solutions

- Decomposition
- Hierarchy
- Partial programs
Action language
First-order reasoning
Hierarchical reinforcement learning
  Learns completions for partial programs
Concurrency
Reflection / meta-object protocols
...

Iliad: A POEM Implementation

- Common Lisp-based programming language
- Full first-order reasoning
  - Operations on logical theories: U(C)NA, domain closure, ...
  - Resolution, hyperresolution, DPLL, etc.
  - Conditional answer extraction
  - Procedural attachment, constraint solving
- Hierarchical reinforcement learning
  - Based on Concurrent ALisp
  - Partial programs
  - Threadwise and temporal state abstraction
  - Hierarchically Optimal Yet Local (HOLY) Q-learning
  - Hierarchically Optimal Recursively Decomposed (HORD) Q-learning
Planned Contents

- Introduction to CALisp/Poem
- Simple TD-learning: bandits
- Flat reinforcement learning: navigation
- Hierarchical reinforcement learning: collecting items individually
- Threadwise decomposition for hierarchical reinforcement learning: learning collaboratively
Choice between $n$ actions
- Reward depends probabilistically on the action choice
- No long-term consequences
- Simplest form of TD-learning

$n$-armed Bandits

search / 1.0 / $\mathcal{N}(0.1, 1.0)$
coll-known / 1.0 / $\mathcal{N}(0.3, 3.0)$
Target: (0 0)

Choices: (N E S W)

Q-values:

#((N (Q -1.8))
   (E (Q -1.8))
   (S (Q -2.25))
   (W (Q 2.76))

Recommended choice is W
(defun simple-robot ()
  (call (nav (target-loc (robot-env))))))

(defun nav (loc)
  (until (equal (robot-loc) loc)
    (with-choice navigate-choice (dir '(N E S W))
      (action navigate-move dir)))))
(defun waste-removal ()
  (loop
    (choose choose-waste-removal-action
      (action sleep 'SLEEP)
      (call (pickup-waste))
      (call (drop-waste))))
)

(defun pickup-waste ()
  (call (nav (waste-source)))
  (action pickup-waste 'PICKUP))
Thank you!

Any Questions?

matthias.hoelzl@ifi.lmu.de