POLITECNICO DI MILANO

Multi-Robot Learning: motivations and approaches

Andrea Bonarini

AI and Robotics Lab
Department of Electronics and Information
Politecnico di Milano

Outline

- Problems in (multi)-robot (reinforcement) learning
  - Stochastic environment
  - Continuous I/O space
  - Effective exploration
  - Delayed rewards

- Some solutions
  - Provide prior information
  - Local exploration
  - Robust learning: the lower bound approach
  - Learning with coarse discretization: piecewise-constant policy

Learning through Reinforcement

The goal of any RL algorithm is to find the **optimal policy**

Q-learning

- The most popular RL algorithm
- Learns the **utility** of executing each action in each state: *Q(s,a)*
- When the robot, starting from state *s* and taking action *a*, reaches *s'* and gets the reward *r*
  - $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$
    - α is a learning rate
    - γ is a discount factor
- Under some conditions *Q(s,a)* **converges** to the optimal *Q*(s,a)*
- The **optimal policy** can be simply derived
  - $\pi(s) = \text{argmax}_a Q^*(s,a)$

What makes Q-learning so popular?

- Is **model-free**
  - implicitly estimated the transition model by direct interaction
- Is **off-policy**
  - learns the optimal policy while following any policy that provides enough exploration
- **Learns on-line**
  - incrementally updates the utility estimates
- Can be **easily and efficiently implemented**
  - the Q-function can be simply stored as a lookup table
  - learning is achieved by performing elementary backup operations

What makes its use in real-world domains so difficult?

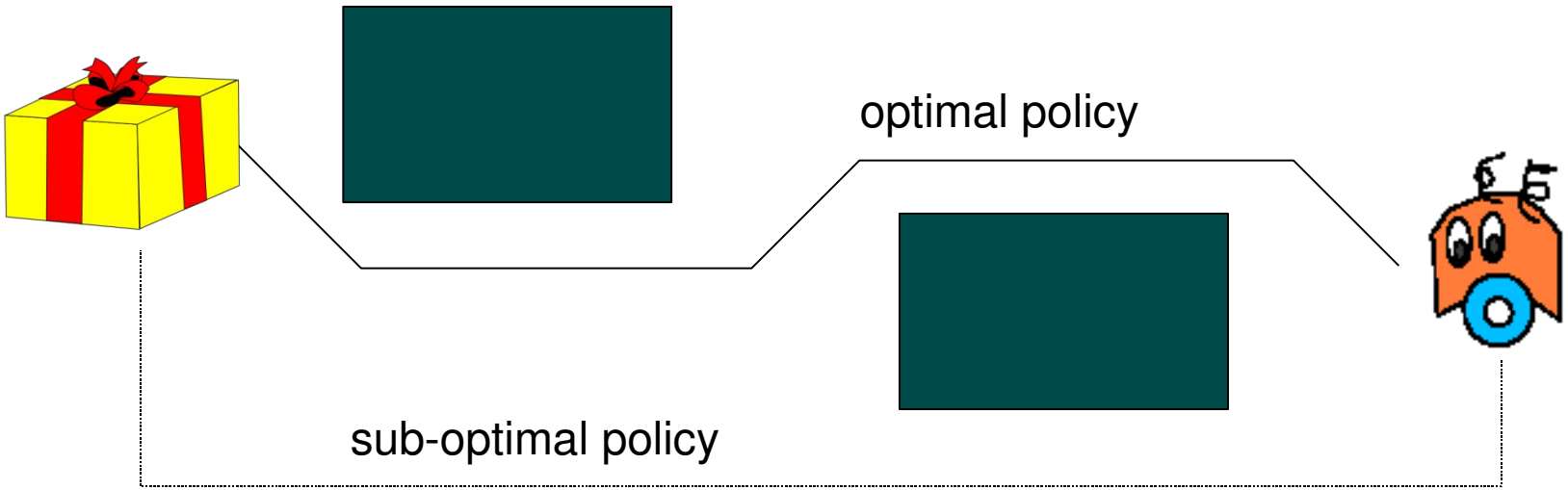
- Learning begins **from scratch**
  - in the first learning trials the robot **wanders blindly**
    - **delayed rewards**
- Traditional RL algorithms are designed to solve **finite** problems
  - robot have both **continuous** perceptions and actuations
    - **discretization**
- Requires **exhaustive exploration** of the state-action space
  - making experience is **expensive** and **dangerous**
- Environment is typically assumed to be **Markovian**
  - robotic tasks are typically partially observable and non-stationary
  - in multi-robot learning also the others are learning
    - **robust** learning methods are required

The Proposed Solution

- We propose to combine several solutions to avoid:
  - learning from scratch
    - by providing prior information (a simple hand-coded solution (ICINCO2008), transfer of solutions learned in easier situations , transfer of samples in batch RL (ICML2008; IFIP2008)
  - continuous actions (NIPS2008) and perceptions
    - aggregation
  - exhaustive exploration
    - by using local exploration
  - effects of uncertainty and non-stationarity
    - by using a new minimax approach
  - oscillatory behaviors
    - by using piecewise-constant policies

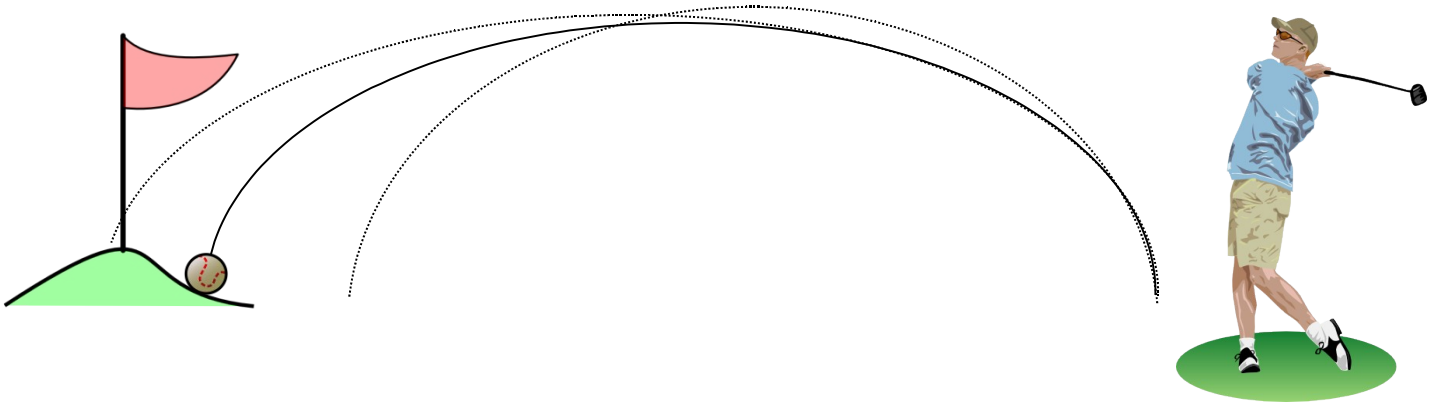
Providing Prior Information

- Learning from **scratch** is **unrealistic** for most robotic tasks
- Although the optimal behavior is unknown, **sub-optimal** controllers may be often implemented with a little effort
- The sub-optimal controller may be used to **initialize** the Q-function [Smart & Kaelbling, 2002] (pessimistic initialization)
- Starting from a sub-optimal policy reduces the effects of delayed rewards



Local Exploration

- Providing prior information avoids to blindly explore the world
- To **improve** the sub-optimal behavior, the robot must **explore alternatives**
- Using exploration strategies, like **ε-greedy**, is too **expensive**
- We propose to use a **local ε-greedy exploration**
  - explorative actions are selected round about the greedy action
  - the sub-optimal controller is **progressively improved**
  - pay attention to **local maxima**



Robust Learning: the Lower-Bound Algorithm

- Robotic domains are typically affected by high **uncertainty** and **non-stationary** phenomena
- Several **Robust RL** approaches have been proposed
  - **minimax principle**
- Heger proposed a Q-learning variant that computes the minimax value [Heger, 1994]: $Q^{k+1}(s,a) = min\left[Q^k(s,a), r + \gamma max_{a'} Q^k(s',a')\right]$
  - requires **optimistic initializations** → exhaustive exploration
- To solve this problem we propose the Lower Bound (LB) Q-learning:

$$Q^{k+1}(s,a) = \begin{cases} r + \gamma max_{a'} Q(s',a'), & if Q^k(s,a) > r + \gamma max_{a'} Q(s',a') \\ (1-\alpha) Q^k(s,a) + \alpha \left(r + \gamma max_{a'} Q(s',a')\right), & otherwise \end{cases}$$

- Q_{LB}-learning
  - does not require optimistic initializations
  - is less sensitive to sporadic non-stationary events

Learning with Coarse Discretizations

- Robotic tasks are characterized by **continuous** state-action spaces
- A common solution is to use **discretizations**
- The state space is partitioned into several **aggregates**
  - the problem becomes **non-Markovian**
  - the learning process has oscillatory behaviors
- We propose to use a piecewise-constant policy
  - when the robot enters in a state, it selects an action and **keeps** it as long as it gets out from that state (or a timeout expires)
  - while the robot remains in the same state, reward is cumulated
  - The problem can be modeled as an SMDP, and the update is

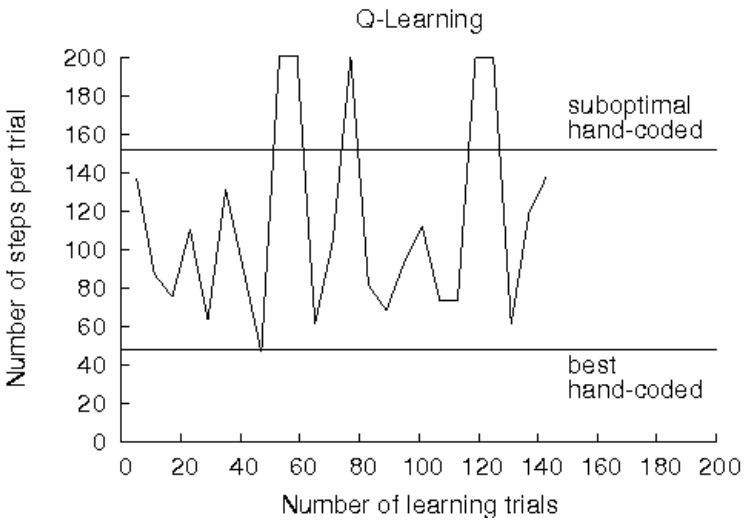
$$Q^{k+1}(s,a) = (1-\alpha)\,Q^k(s,a) + \alpha\left(\sum_{i=1}^{N} \gamma^{i-1} r_i + \gamma^N \max_{a'} Q^k(s',a')\right)$$

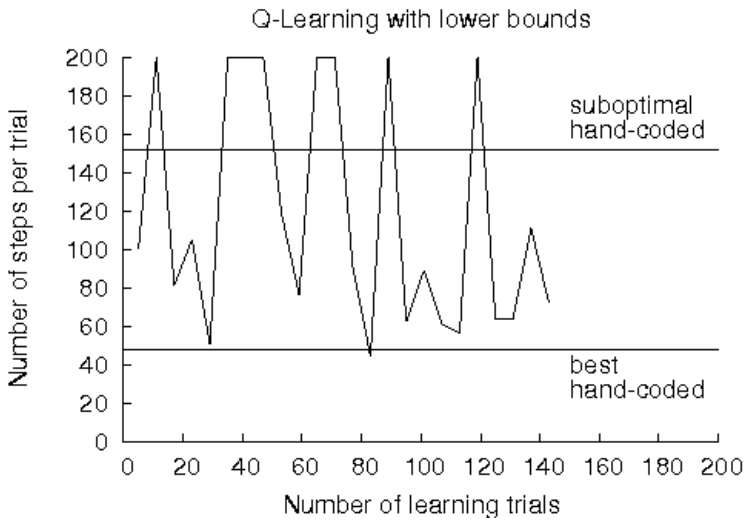
Experiments with a Real Robot

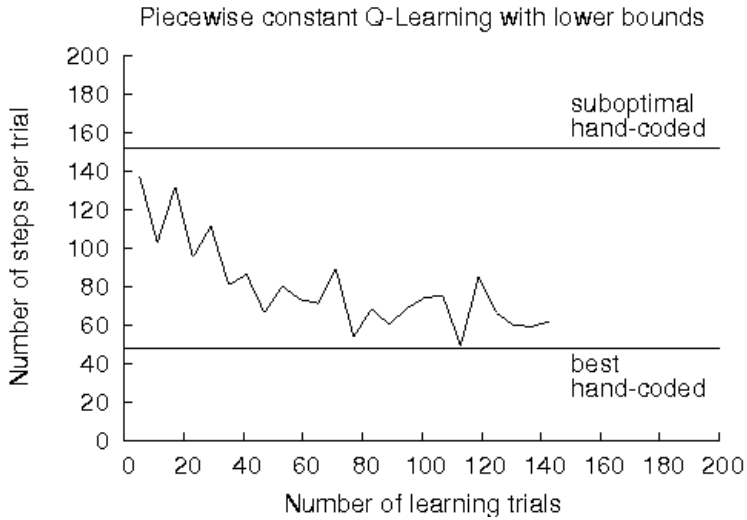
- The proposed solutions have been tested on a **real robot**
- The robot participates to the RoboCup competition with MRT
  - omni-directional wheels and a 360° vision sensor
- Task faced: **Go-to-ball**
  - state space (two vars)
    - ball angle and ball distance
    - discretized in 120 aggregates
  - action space (three vars)
    - module and angle of tangential speed
    - rotational speed
    - 1,296 available joint actions
  - reward function
    - positive on reaching the ball, null otherwise

Let's see the robot in action...

Experimental results





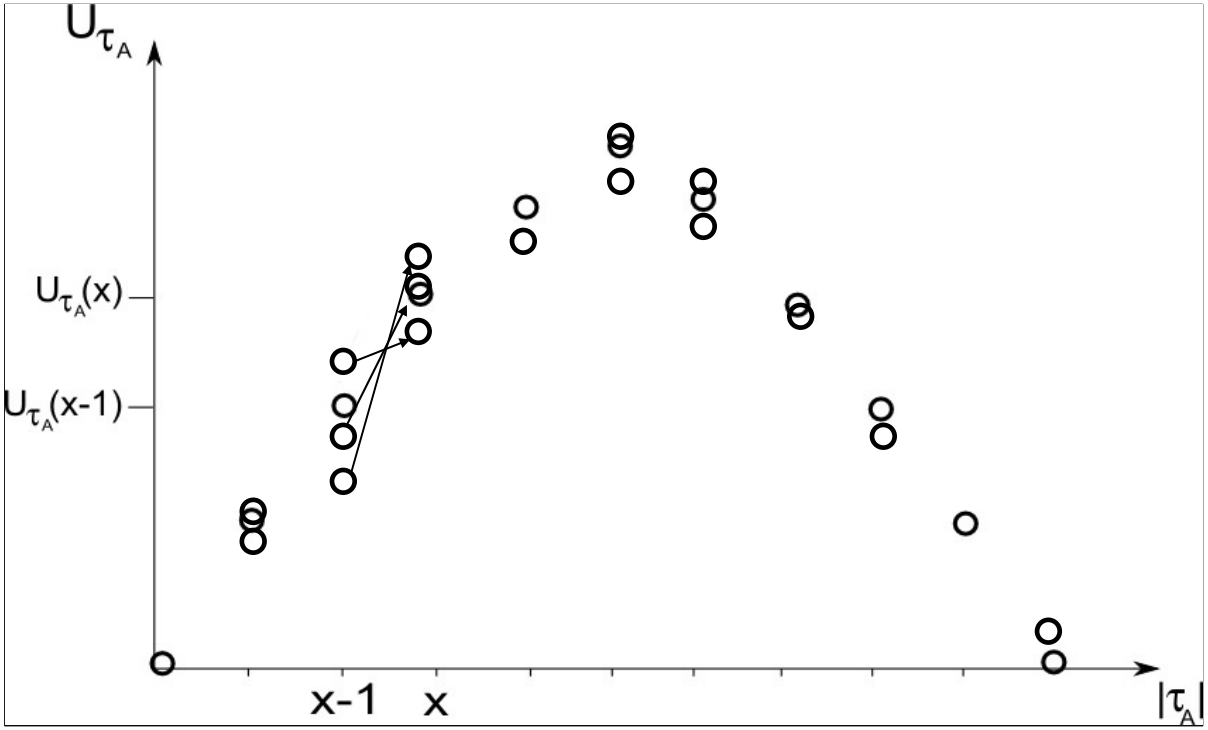


Learning coordination

- Each robot learns how to coordinate its activity with other's, i.e., which task to take -> task allocation
- Goal: maximize the global utility, i.e., the sum of the utilities of the tasks
- “Classical” approach: COIN (Wolpert&Tumer 1999), a RL distributed approach to learn task allocation without communication
- Reinforcement “Wonderful Life”: the increment of utility caused by the fact that an agent takes a task
- Problems with COIN: task independence is assumed (but usually they aren't), perfect knowledge about the utility functions is needed (but usually they are not known and stochastic

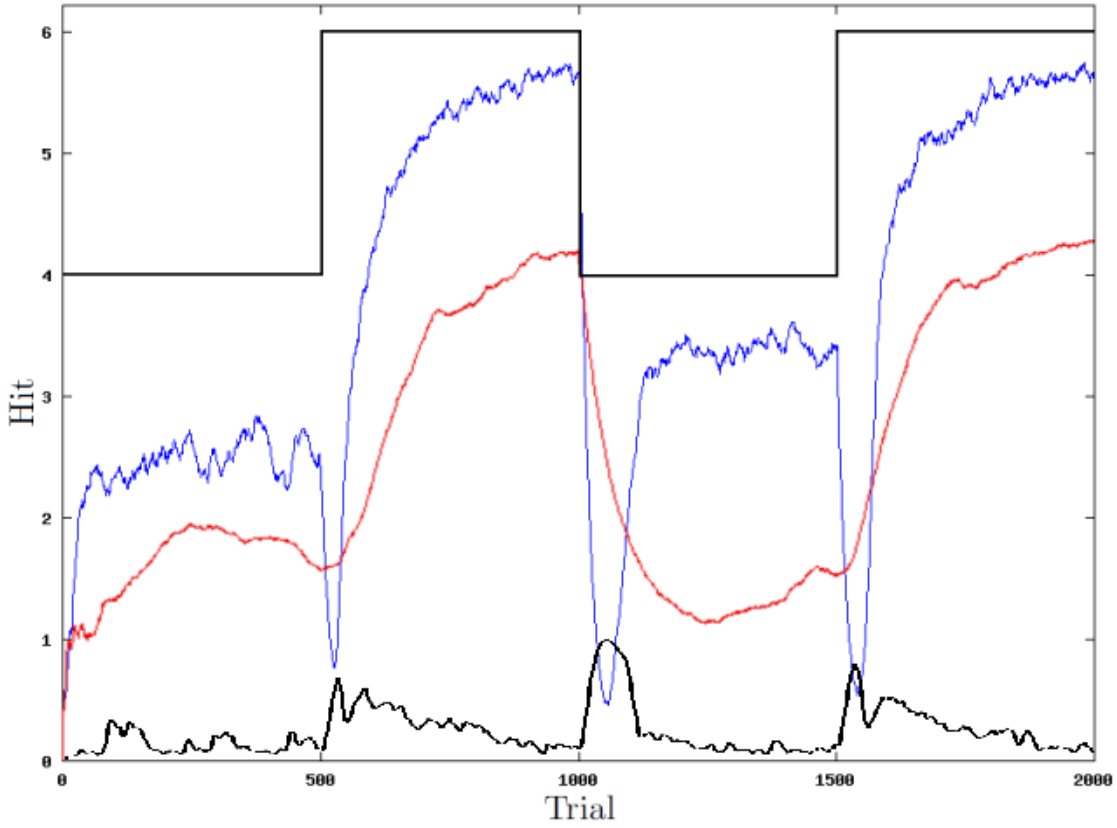
To use COIN in real multi-robot tasks

- Estimate the utility function



To face dynamical environments

- Identify changes in the environment, and eventually increase the learning rate



Conclusions

- Although **learning** capabilities are needed to achieve full **autonomy**, the application of direct learning methods in robotic applications rises several issues
- We have proposed some approaches that combines solutions to several different problems
- The preliminary results obtained on both agent and robotic tasks show that the proposals effectively improve the performance of basic RL algorithms

Thank you for your attention

Any question?