Reinforcement Learning and Policy Reuse

Manuela M. Veloso

10-601, Fall 2017

Readings:

- Reinforcement Learning: An Introduction, R. Sutton and A. Barto
- Probabilistic policy reuse in a reinforcement learning agent, Fernando Fernandez and Manuela Veloso. In *Proceedings of AAMAS'06.*

Learning

- Learning from experience
- Supervised learning
  - Labeled examples
- Reward/reinforcement
  - Something good/bad (positive/negative reward) happens
  - An agent gets reward as part of the “input” percept, but it is “programmed” to understand it as reward.
  - Reinforcement extensively studied by animal psychologists.

Online Learning Approaches

- Capabilities
  - Execute actions in world
  - Observe state of world

- Two Learning Approaches
  - Model-based
  - Model-free

Model-Based Reinforcement Learning

- Approach
  - Learn the MDP
  - Solve the MDP to determine optimal policy
- Appropriate when model is unknown, but small enough to solve feasibly

Learning the MDP

- Estimate the rewards and transition distributions
  - Try every action some number of times
  - Keep counts (frequentist approach)
    - $R(s,a) = R^a_s / N^a_s$
    - $T(s',a,s) = N^a_{s,s'} / N^a_s$
  - Solve using value or policy iteration

- Iterative Learning and Action
  - Maintain statistics incrementally
  - Solve the model periodically

Model-Free Reinforcement Learning

- Learn policy mapping ***directly***

- Appropriate when model is too large to store, solve, or learn
  - Do not need to try every state/action in order to get good policy
  - Converges to optimal policy

Learn Value Function

- Learn the evaluation function $V^{\pi *}$ (i.e. V^*)
- Select the optimal action from any state s, i.e., have an optimal policy, by using V^* with one step lookahead:

$$\pi^*(s) = \arg\max_a \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$$

- But reward and transition functions are unknown

Q Function

- Define new function very similar to V^*

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

Learn Q function – Q-learning

- If agent learns Q, it can choose optimal action even without knowing δ or r

$$\pi^*(s) = \arg\max_a \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$$

$$\pi^*(s) = \arg\max_a Q(s,a)$$

Training Rule to Learn Q (Deterministic Example)

Let $\hat{Q}$ denote current approximation to Q.

Then Q-learning uses the following **training rule**:

$$\boxed{\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')}$$

where s' is the state resulting from applying action a in state s, and r is the reward that is returned.

Nondeterministic Case

- Q learning in nondeterministic worlds
  - Redefine V, Q by taking expected values:

$$
\begin{aligned}
V^{\pi}(s) &\equiv E\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots\right] \\
&\equiv E\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right]
\end{aligned}
$$

$$
Q(s,a) \equiv E\left[r(s,a) + \gamma V^{*}(\delta(s,a))\right]
$$

Nondeterministic Case

- Q learning training rule:

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n\left[r + \gamma \max_{a'} \hat{Q}_{n-1}(s',a')\right],$$

where $\alpha_n = \frac{1}{1+visits_n(s,a)}$, and $s' = \delta(s,a)$.

$\hat{Q}$ still converges to Q^* (Watkins and Dayan, 1992)

Exploration vs Exploitation

- Tension between learning optimal strategy and using what you know, so far, to maximize expected reward
  - Convergence theorem depends on visiting each state sufficient number of times
  - Typically use reinforcement learning while performing tasks

Exploration policy

- *Wacky approach*: act randomly in hopes of eventually exploring entire environment
- *Greedy approach*: act to maximize utility using current estimate
- *Balanced approach*: act "more" wacky when agent has not much knowledge of environment and "more" greedy when the agent has acted in the environment longer
- One-armed bandit problems

Exploration Strategies

- ε-greedy
  - Exploit with probability 1-ε
  - Choose remaining actions uniformly
  - Adjust ε as learning continues
- Boltzman
  - Choose action with probability

$$p = \frac{e^{Q(s,a)/t}}{\sum_{a'} e^{Q(s,a')/t}}$$

All methods sensitive to parameter choices and changes

Policy Reuse

- Impact of change of reward function
  - Does not want to learn from scratch
- Transfer learning
  - Learn macros of the MPD – options
  - Value function transfer
  - Exploration bias
- Reuse complete policies

Episodes

- MDP with absorbing goal states
  - Transition probability from a goal state to the same goal state is 1 (therefore to any other state is 0)
- Episode:
  - Start in random state, end in absorbing state
- Reward per episode (K episodes, H steps each):

$$W = \frac{1}{K} \sum_{k=0}^{K} \sum_{h=0}^{H} \gamma^h r_{k,h} \tag{1}$$

where γ $(0 \leq \gamma \leq 1)$ reduces the importance of future rewards, and $r_{k,h}$ defines the immediate reward obtained in the step h of the episode k, in a total of K episodes.

Q-Learning

Q-Learning (K, H, γ, α).

Initialize $Q(s, a)$, $\forall s \in \mathcal{S}, a \in \mathcal{A}$

For $k = 1$ to K

    Set the initial state, s, randomly.

    for $h = 1$ to H

        Select an action a and execute it

        Receive current state s', and reward, $r_{k,h}$

        $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r_{k,h} + \gamma \max_{a'} Q(s', a')]$

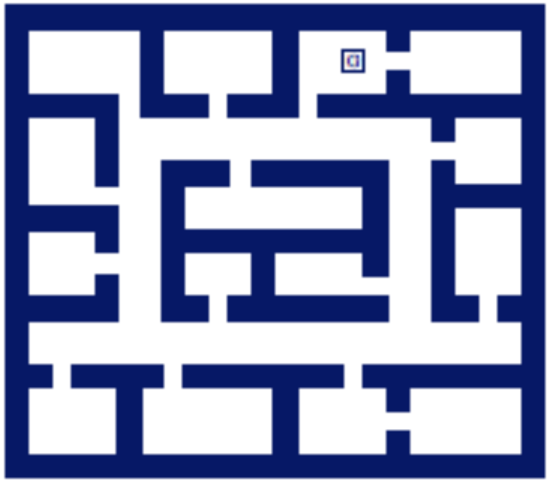
        Set $s \leftarrow s'$

$W = \frac{1}{K}\sum_{k=0}^{K}\sum_{h=0}^{H}\gamma^h r_{k,h}$

Return W, $Q(s, a)$ and Π

Experimental Domain

- Continuous state space x, y (optimal discretization)
- Size: 24×21
- Discrete set of actions: Go north, south, east and west, each step of size 1
- Noise in actuators
- Obstacle avoidance system
- Each episode starts in a random initial position

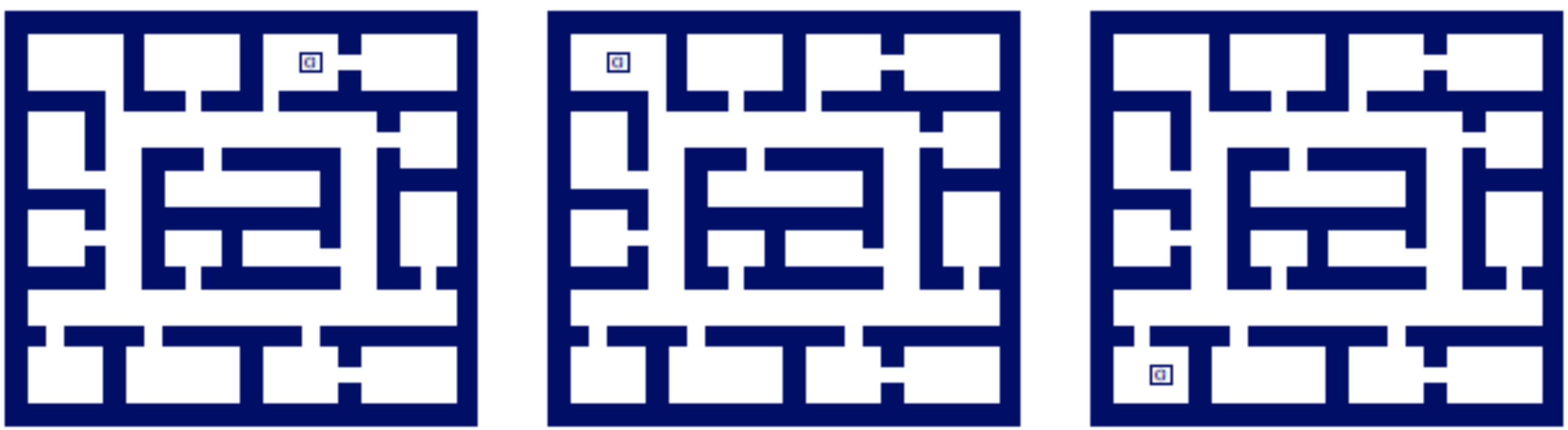


Domains and Tasks

A **domain** $\mathcal{D}$ is defined as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T} >$, where $\mathcal{S}$ is the set of all possible states; $\mathcal{A}$ is the set of all possible actions; and $\mathcal{T}$ is a state transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \Re$

A **task** Ω is defined as a tuple $< \mathcal{D}, \mathcal{R}_\Omega >$, where $\mathcal{D}$ is a domain; and $\mathcal{R}_\Omega$ is the reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$

An **action policy** Π_Ω to solve a task Ω is a function $\Pi_\Omega : \mathcal{S} \rightarrow \mathcal{A}$.



(a) Task Ω_1 (b) Task Ω_2 c) Task Ω_3

Policy Library and Reuse

- **Policy Reuse**:
  - ⋆ We need to solve the task Ω, i.e. learn Π_Ω
  - ⋆ We have previously solved the set of tasks $\{\Omega_1, \ldots, \Omega_n\}$ so we have a Policy Library composed of the n policies that solve them respectively, say $L = \{\Pi_1, \ldots, \Pi_n\}$
  - ⋆ How can we use the policy library, L, to learn the new policy, Π_Ω?

π-Reuse Exploration

Need to solve a task Ω, i.e. learn Π_{new}.

Have a Policy Library, say $L = \{\Pi_1, \ldots, \Pi_n\}$

Let's assume that there is a supervisor who, given Ω, tells us which is the most similar policy, say Π_{past}, to Π_{new}. Thus, we know that the policy to reuse is Π_{past}.

Integrate the past policy as a probabilistic bias in the exploration strategy of the new learning process

Define probabilities for exploiting the past policy, perform random exploration, or exploit the ongoing policy

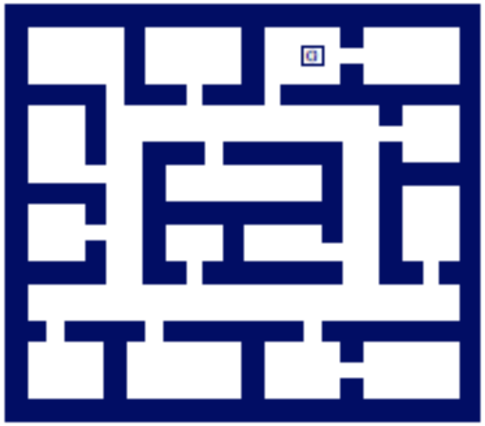
⋆ Select $a = \begin{cases} \Pi_{past}(s) & \text{w/prob. } \psi \\ \Pi_{new}(s)) & \text{w/prob. } (1-\psi)\epsilon \\ Random & \text{w/prob. } (1-\psi)(1-\epsilon) \end{cases}$

π-Reuse Policy Learning

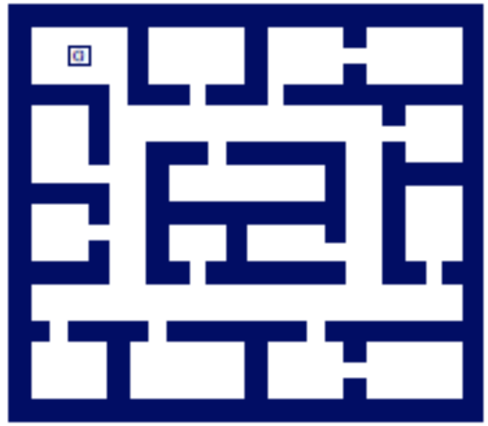
π-reuse $(\Pi_{past}, K, H, \psi, \upsilon, \gamma, \alpha)$.

Initialize $Q^{\Pi_{new}}(s,a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
For $k = 1$ to K
  Set the initial state, s, randomly.
  Set $\psi_1 \leftarrow \psi$
  for $h = 1$ to H
    With a probability of ψ_h, $a = \Pi_{past}(s)$
    With a probability of $1 - \psi_h$, $a = \epsilon\text{-greedy}(\Pi_{new}(s))$
    Receive current state s', and reward, $r_{k,h}$
    Update $Q^{\Pi_{new}}(s,a)$, and therefore, Π_{new}, using the Q-Learning update function:
      $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \operatorname{máx}_{a'} Q(s',a')]$
    Set $\psi_{h+1} \leftarrow \psi_h \upsilon$
    Set $s \leftarrow s'$
$W = \frac{1}{K}\sum_{k=0}^{K}\sum_{h=0}^{H}\gamma^h r_{k,h}$
Return W, $Q^{\Pi_{new}}(s,a)$ and Π_{new}

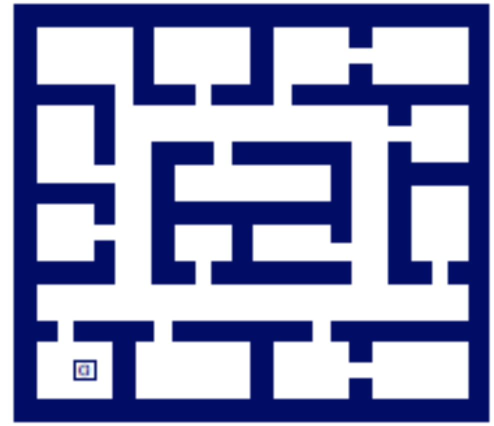
Experimental Results



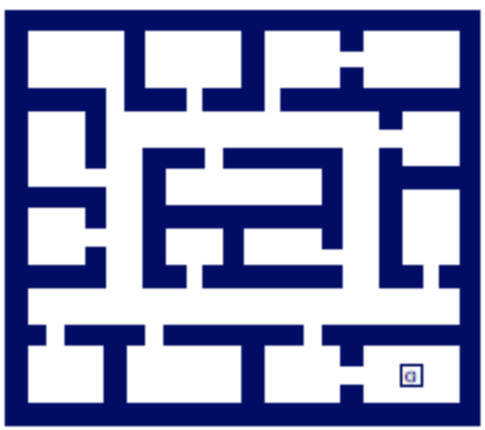
(a) Task Ω_1



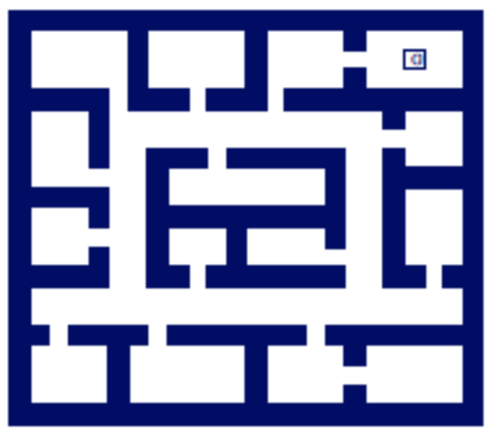
(b) Task Ω_2



c) Task Ω_3



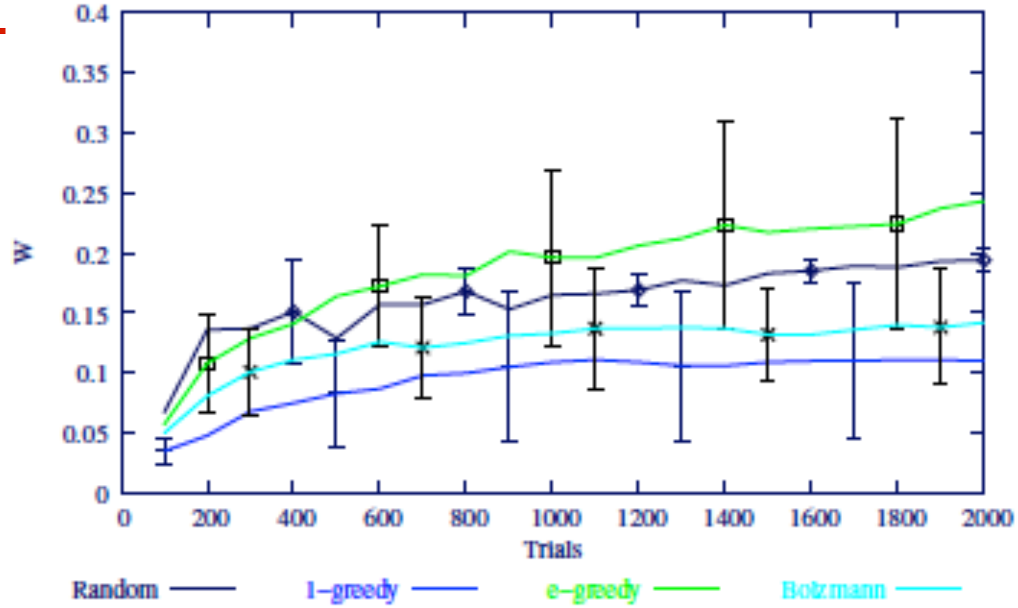
(d) Task Ω_4



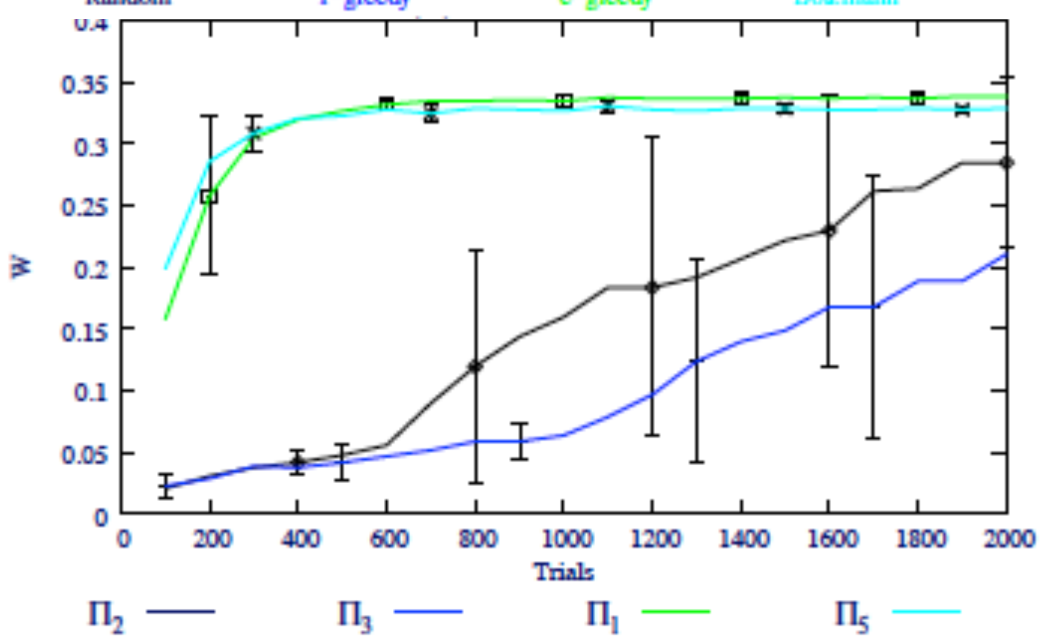
(e) Task Ω_5

(f) Task Ω

Results



0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
W
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Trials
Random
1-greedy
e-greedy
Boltzmann



0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
W
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Trials
Π_2
Π_3
Π_1
Π_5

Policy Reuse Using a Policy Library

- What if a “good” similar policy is not given
- Interestingly, the pi-reuse strategy also contributes a *similarity metric* between policies:
  - The gain Wi obtained while executing the pi-reuse exploration strategy, reusing the past policy i.
- Wi is an estimation of how similar the policy i is to the new one!
- The set of Wi values for each of the policies in the library is unknown a priori, but it can be estimated on-line while the new policy is computed in the different episodes.

Using a Library of Policies

1. Given the set of policies composed of $L \cup \{\Pi_\Omega\} = \{\Pi_\Omega, \Pi_1, \ldots, \Pi_n\}$, what policy is followed in each episode?
   - ⋆ $P(\Pi_j) = \frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}}$
2. Once a policy, Π_k is selected, what exploration strategy is followed?
   - ⋆ Depends on the policy:
     - ∗ If $\Pi_k \neq \Pi_\Omega$, then $\pi - reuse$
     - ∗ If $\Pi_k = \Pi_\Omega$, then greedy.
3. How is W_j computed?
   - ⋆ On line with the learning of the new policy

PRQ-Learning.

PRQ-Learning (Ω, L, K, H)

- Given:

(1) A new task Ω we want to solve.
(2) A Policy Library $L = \{\Pi_1, \ldots, \Pi_n\}$.
(3) A maximum number of episodes to execute, K.
(4) A maximum number of steps per episode, H.

- Initialize:

(1) $Q_\Omega(s, a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$.
(2) $W_\Omega = W_i = 0$, for $i = 1, \ldots, n$.

- For $k = 1$ to K do

– Choose an action policy, Π_k, assigning to each policy the probability of being selected computed by the following equation:

$$P(\Pi_j) = \frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}}$$

where W_0 is set to W_Ω.

– Execute the learning episode k.

If $\Pi_k = \Pi_\Omega$, execute a Q-Learning episode following a fully greedy strategy.

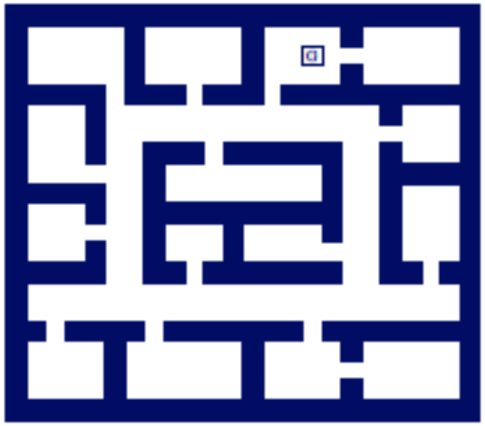
Otherwise, call π-reuse (Π_k, 1, H, ψ, υ).

In any case, receive the reward obtained in that episode, say R, and the updated Q function, $Q_\Omega(s, a)$.

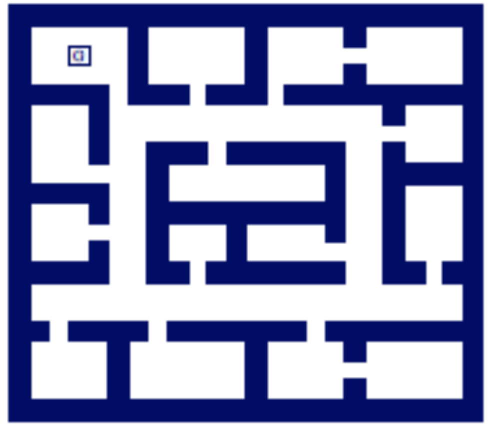
– Recompute W_k using R.

- Return the policy derived from $Q_\Omega(s, a)$.

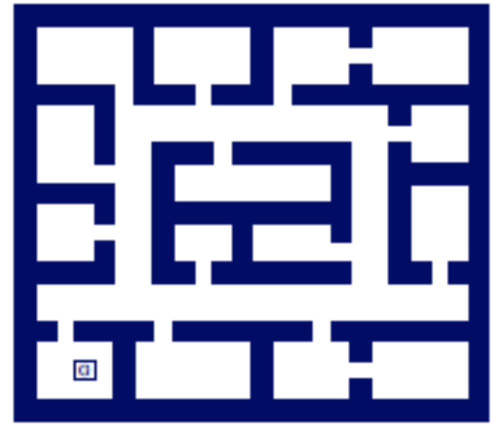
Experimental Results



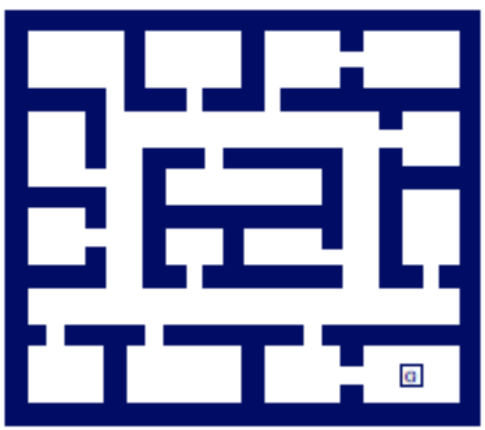
(a) Task Ω_1



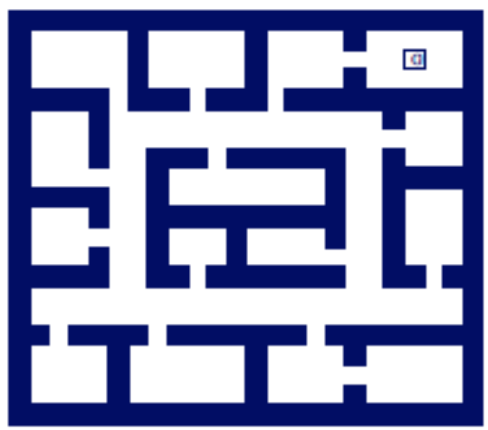
(b) Task Ω_2



c) Task Ω_3



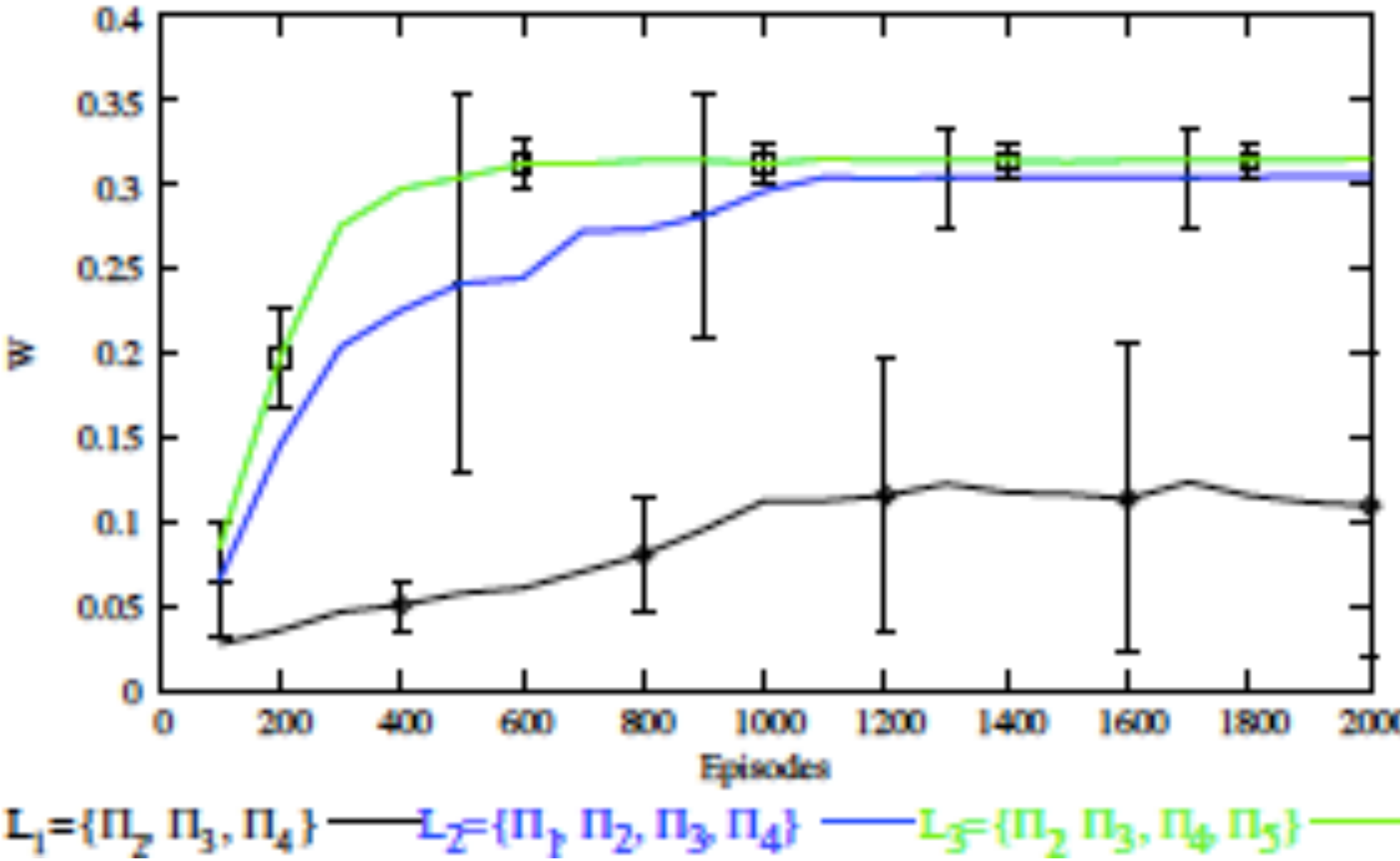
(d) Task Ω_4



(e) Task Ω_5

(f) Task Ω

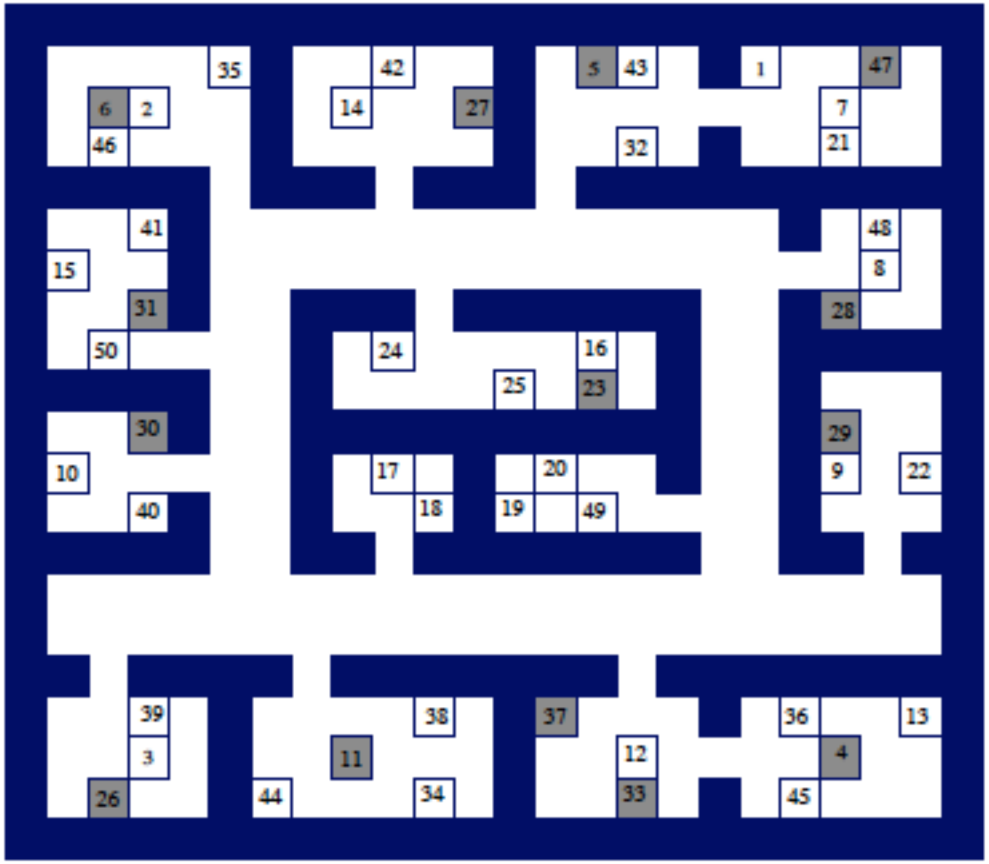
Results



0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
W
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Episodes
$L_1=\{\Pi_2, \Pi_3, \Pi_4\}$
$L_2=\{\Pi_1, \Pi_2, \Pi_3, \Pi_4\}$
$L_3=\{\Pi_2, \Pi_3, \Pi_4, \Pi_5\}$

Learning a Policy Library

- Similarity between policies can be learned
- Gain of using each policy
- Explore different policies
- Learn domain structure: “eigen” policies



Summary

- An exploration strategy to bias the learning of the new task with a given past policy
  - π-reuse exploration
- An algorithm that discriminates among several past policies to decide which is best to reuse
  - PRQ-learning algorithm
- Mentioned: A method to incrementally build the policy library