



## Storage

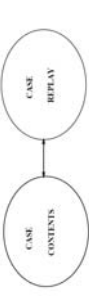
- Efficient indexing
  - footprinting initial state
  - interacting goals
- Initial state is organized as a discrimination network.

Simmons, Veloso, Carnegie Mellon

7

15-887  
Fall/2010

## Generation and Replay



Generation and replay share representational language

- Replay involves:
- a complete reinterpretation of the justification structures in the new context
  - the development of appropriate actions to be taken when transformed justifications are no longer valid.

Simmons, Veloso, Carnegie Mellon

9

15-887  
Fall/2010

## Advantages of Replay

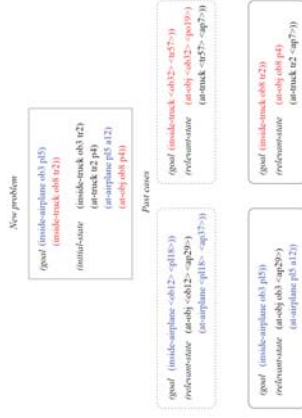
- Proposal and validation of choices versus generation and search of alternatives
- Reduction of the branching factor
  - past failed alternatives are pruned by validating the past failure records;
  - if needed, PRODIGY/ANALOGY backtracks also in the guiding cases and uses information on failure to make more informed backtracking decisions.
- Subgoal links identify the subparts of the case to replay -- the steps that are not part of the active goals are skipped.

Simmons, Veloso, Carnegie Mellon

11

15-887  
Fall/2010

## Retrieval of Past Similar Problems



Simmons, Veloso, Carnegie Mellon

8

15-887  
Fall/2010

## Reuse of Annotated Experience

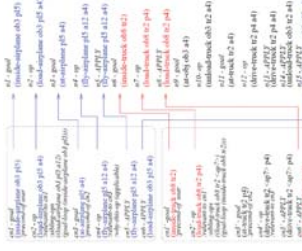
- |                      |                         |
|----------------------|-------------------------|
| Chosen step          | Proposed step           |
| Subgoal links        | Search direction        |
| Sibling alternatives | Proposed sibling steps  |
| Record of failures   | Pruning of alternatives |
| Additional reasons   | Additional control      |
- Extend case when extra planning is needed  
Reduce case when past planning is not needed

Simmons, Veloso, Carnegie Mellon

10

15-887  
Fall/2010

## Replay of Multiple Episodes

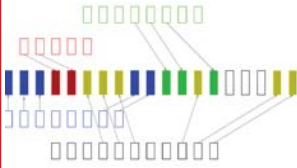


Simmons, Veloso, Carnegie Mellon

12

15-887  
Fall/2010

## Replaying Multiple Cases



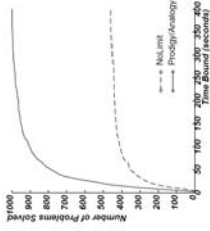
Simmons, Veloso, Carnegie Mellon

13

15-887  
Fall2010

## Experiments

- Several different domains, including logistics transportation
- Solvability horizon of generative planner is greatly increased due to the integrated replay of planning cases.



Simmons, Veloso, Carnegie Mellon

15

15-887  
Fall2010

## Example - Route Planning



A particular threshold situation:  
relevant cases for bnd



bnd solved by Analogy:  
red shows the extra planning done.  
the three cases are merged with extra planning.  
irrelevant parts of the cases are not used.

(Hahn & Veloso - ICCBR'05; Hahn, Shewchuk, & Veloso - JETAI'97)

Simmons, Veloso, Carnegie Mellon

17

15-887  
Fall2010

## Analogical Replay Of Multiple Planning Cases

1. Terminate if the goal is satisfied in the state.
2. Choose a guiding case. If a failure, then backtrack and set back guiding cases.
3. If a goal is chosen, then
  - 3.1. Validate the goal justifications. If not validated, go to step 2.
  - 3.2. Create a new goal node; link it to the case node. Advance the case.
  - 3.3. Select the operator chosen in the case.
  - 3.4. Validate the operator and bindings choices. If not validated, base-level plan for the goal. Go to step 2.
  - 3.5. Link the new operator node to the case node. Advance the case. Go to step 2.
4. If an applicable operator is chosen, then
  - 4.1. Check if it can be applied in the current state. If it cannot, go to step 2.
  - 4.2. Link the new applied operator node to the case node. Advance the case. Apply the operator. Go to step 1.

Simmons, Veloso, Carnegie Mellon

14

15-887  
Fall2010

## Application Domain: Route Planning

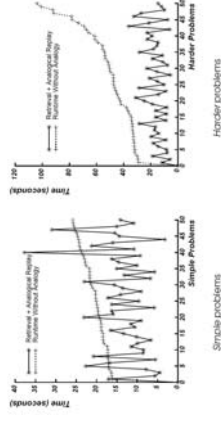
- Routes are accumulated in a case library.
- Routes are abstracted and indexed according to situational parameters, such as: time of the day, day of the week, and driver.
- Geometric features are used by the similarity metric used at retrieval time.
- Multiple routes are merged at planning time.
- Planning cases are integrated with generative planning.
- Relevant parts of the cases are validated, pursued and merged.
- Generative planner does any extra planning work needed to merge the planning cases.

Simmons, Veloso, Carnegie Mellon

16

15-887  
Fall2010

## Retrieval plus Replay Time



Simmons, Veloso, Carnegie Mellon

18

15-887  
Fall2010

# Replaying Plans - Summary

Characteristics of learning by analogical reasoning in

PRODIGY/ANALOGY:

- The strategy-level learning process is cast as the automation of the complete cycle of
  - constructing,
  - storing,
  - retrieving,
  - and replaying problem solving episodes.
- No substantial effort invested in deriving general rules of behavior to apply to individual decisions.
- Learned knowledge is flexibly applied to new situations, i.e., even if only a partial match exists among past and new problems.

## Introduction

### Reinforcement Learning

- Reinforcement Learning problem (defined as an MDP):
  - \* the set of all the possible states,  $S$ ,
  - \* the set of all the possible actions,  $\mathcal{A}$ ,
  - \* an unknown state transition function,  $\mathcal{T} : S \times \mathcal{A} \times S \rightarrow \mathbb{R}$ , and
  - \* an unknown reward function,  $\mathcal{R} : S \times \mathcal{A} \rightarrow \mathbb{R}$ .
- Goal: learn the action policy  $\Pi : S \rightarrow \mathcal{A}$  that maximizes the expected average reward, or gain (we assume episodic tasks with absorbing goal states and random initial positions):

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

- \*  $K$ : number of episodes
- \*  $H$ : number of steps per episode

## Introduction

### Transfer Learning in RL

- Transfer learning:
  - \* Learn a new abstraction of the MDP: options (Sutton, Precup and Singh, 1999), skills (Thrun and Schwartz, 1995), . . .
  - \* Value function transfer (Taylor and Stone, 2005), (Carroll and Peterson 2002), . . .
  - \* Exploration bias: advice rules (Maclin et al, 2005), imitation (Price and Boutiller, 2003), . . .
- There are still some questions in the air:
  - \* How can we reuse complete policies?
  - \* How can we decide whether the knowledge acquired in past learning processes is useful for a new one?
  - \* How much knowledge do we need to store/use in the lifelong term?

## Policy Reuse

Manuela Veloso

Joint work with, and thanks to,  
Fernando Fernandez

Planning, Execution, and Learning  
Fall 2010

## Introduction

### Q-Learning (Watkins, 1989)

---

Q-Learning ( $K, H, \gamma, \alpha$ ).

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

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  $s = s'$

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

Return  $W, Q(s, a)$  and  $\Pi$

---

## Introduction

### Policy Reuse

- A domain  $\mathcal{D}$  is defined as a tuple  $\langle S, \mathcal{A}, \mathcal{T} \rangle$ , where  $S$  is the set of all possible states;  $\mathcal{A}$  is the set of all possible actions; and  $\mathcal{T}$  is a state transition function,  $\mathcal{T} : S \times \mathcal{A} \times S \rightarrow \mathbb{R}$
- A task  $\Omega$  is defined as a tuple  $\langle \mathcal{D}, \mathcal{R}_\Omega, \gamma \rangle$ , where  $\mathcal{D}$  is a domain; and  $\mathcal{R}_\Omega$  is the reward function,  $\mathcal{R} : S \times \mathcal{A} \rightarrow \mathbb{R}$
- An action policy  $\Pi_\Omega$  to solve a task  $\Omega$  is a function  $\Pi_\Omega : S \rightarrow \mathcal{A}$ .
- Policy Reuse:
  - \* We need to solve the task  $\Omega$ , i.e. learn  $\Pi_\Omega$
  - \* We have previously solved the set of tasks  $\{\Omega_1, \dots, \Omega_n\}$  so we have a Policy Library composed of the  $n$  policies that solve them respectively, say  $L = \{\Pi_1, \dots, \Pi_n\}$
  - \* How can we use the policy library,  $L$ , to learn the new policy,  $\Pi_\Omega$ ?

### Introduction

## Policy Reuse Components

- An exploration strategy to bias the learning of the new task with one of the past policies in each episode ( $\pi$ -reuse exploration strategy)
- An algorithm that discriminate among several past policies to decide which of them to reuse (PRQ-Learning algorithm)
- A method to incrementally build the policy library (PLPR algorithm)

Fernández and Veloso, December 2005.

8

### Reusing a Past Policy

## $\pi$ -reuse Exploration Strategy

---

$\pi$ -reuse ( $\Pi_{reuse}, K, H, \psi, \psi', \gamma, \alpha$ ).

Initialize  $Q_{reuse}(s, a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$

For  $h = 1$  to  $K$

Set the initial state,  $s$ , randomly.

Set  $\psi_1 \leftarrow \psi$

for  $h = 1$  to  $H$

With a probability of  $\psi_h, a = \Pi_{reuse}(s)$

With a probability of  $1 - \psi_h, a = \epsilon$ -greedy( $\Pi_{reuse}(s)$ )

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

Update  $Q_{reuse}(s, a)$ , and therefore,  $\Pi_{reuse}$ , using the Q-Learning update function:

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

Set  $\psi_{h+1} \leftarrow \psi_h \psi'$

Set  $s \leftarrow s'$

$$W = \frac{1}{K} \sum_{h=0}^{K-1} \sum_{a=0}^H \gamma^h R_{h,h}$$

Return  $W, Q_{reuse}(s, a)$  and  $\Pi_{reuse}$

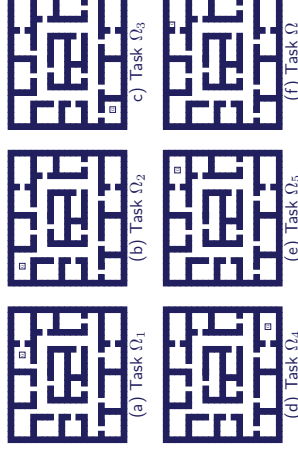
---

Fernández and Veloso, December 2005.

11

### Reusing a Past Policy

## Tasks



Fernández and Veloso, December 2005.

13

### Reusing a Past Policy

## $\pi$ -reuse Exploration Strategy

- Need to solve a task  $\Omega$ , i.e. learn  $\Pi_{reuse}$ .
- Have a Policy Library, say  $L = \{\Pi_1, \dots, \Pi_n\}$
- Let's assume that there is a supervisor who, given  $\Omega$ , tells us which is the most similar policy, say  $\Pi_{reuse}$ , to  $\Pi_{reuse}$ . Thus, we know that the policy to reuse is  $\Pi_{reuse}$ .
- 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

$$\star \text{ Select } a = \begin{cases} \Pi_{reuse}(s) & w/\text{prob.} \\ \text{Random}(s) & w/\text{prob.} \end{cases} \frac{w/\text{prob.}}{w/\text{prob.} + w/\text{prob.}} \frac{(1 - \psi)\epsilon}{(1 - \psi)\epsilon + w/\text{prob.}} (1 - \psi)(1 - \epsilon)$$

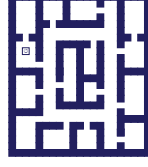
Fernández and Veloso, December 2005.

10

### Reusing a Past Policy

## Domain

- Continuous state space  $x, y$  (optimal discretization)
- Size:  $24 \times 21$
- 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



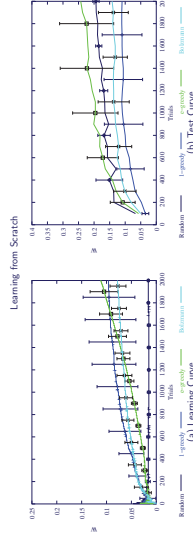
Fernández and Veloso, December 2005.

12

### Reusing a Past Policy

## Results: Learning from Scratch

- Parameters:  $K = 2000, H = 100, \gamma = 0.95, \alpha = 0.05$
- Tests over 1000 episodes
- Average of 10 different learning and test executions

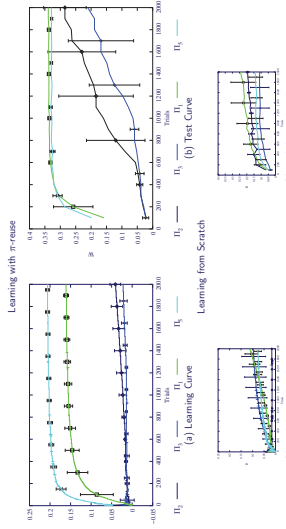


Fernández and Veloso, December 2005.

14

Results: Learning with  $\pi$ -reuse

- Parameters: the same as before, with  $\psi = 1$  and  $v = 0.95$



Fernández and Veloso, December 2005.

15

Fernández and Veloso, December 2005.

## Questions

- Given the set of policies composed of  $L \cup \{\Pi_\Omega\} = \{\Pi_\Omega, \Pi_1, \dots, \Pi_n\}$ , what policy is followed in each episode?
  - \*  $P(\Pi_j) = \frac{e^{-W_j}}{\sum_{p=0}^n e^{-W_p}}$

- Once a policy  $\Pi_k$  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.

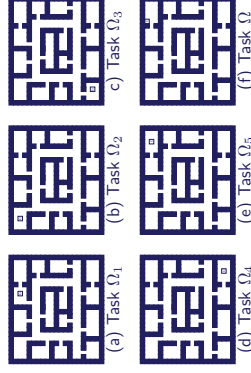
- How is  $W_j$  computed?
  - \* On line with the learning of the new policy

18

Fernández and Veloso, December 2005.

## Experiments

- Reuse:  $L_1 = \{\Pi_2, \Pi_3, \Pi_4\}$ ,  $L_2 = \{\Pi_1, \Pi_2, \Pi_3, \Pi_4\}$  and  $L_3 = \{\Pi_5, \Pi_3, \Pi_4, \Pi_5\}$



18

Fernández and Veloso, December 2005.

## Reuse Gain

- Given:
  - \* a policy  $\Pi_i$  that solves a task  $\Omega_i = \langle \mathcal{D}, R_i \rangle$
  - \* a new task  $\Omega = \langle \mathcal{D}, R_\Omega \rangle$ :
    - the **reuse gain** of the policy  $\Pi_i$  on the task  $\Omega$ , say  $W_i$ , is the gain obtained when applying the  $\pi$ -reuse exploration strategy with the policy  $\Pi_i$  to learn the policy  $\Pi_\Omega$ .
  - \* The most useful policy to reuse, say  $\Pi_k$ , from a Policy Library,  $L = \{\Pi_1, \dots, \Pi_n\}$ , is the one that maximizes the Reuse Gain:
 
$$\Pi_k = \text{argmax}_i (W_i), i = 1, \dots, n \quad (1)$$

Fernández and Veloso, December 2005.

16

Fernández and Veloso, December 2005.

## PRQ-Learning Algorithm

PRQ( $\Omega, L, \tau, \Delta r, K, H, \psi, v, \gamma, \alpha$ )

- Given:
  - A new task  $\Omega$  we want to solve
  - A Policy Library  $L = \{\Pi_1, \dots, \Pi_n\}$
  - $\tau$ , and an incremental size,  $\Delta r$ , for the Boltzmann policy selection strategy
  - A maximum number of episodes to execute,
  - A maximum number of steps per episode,  $H$
  - The parameters  $\psi$  and  $v$  for the  $\pi$ -exploration strategy
  - The parameters  $\gamma$  and  $\alpha$  for the Q-learning strategy in a policy
  - A  $\epsilon$  parameter, for  $\epsilon$ -greedy action selection
- Initialize:
  - $Q_\Omega(s, a) = 0, \forall s \in S, a \in A$
  - Initialize  $W_\Omega$  to 0
  - Initialize  $W_j$  to 0
  - Initialize  $L_j$  to the number of episodes where policy  $\Pi_j$  has been chosen,  $L_j = 0$
  - Set  $\tau = \tau + \Delta r$
- If  $L_j$  has been chosen,  $L_j = 0, \forall j = 1, \dots, n$

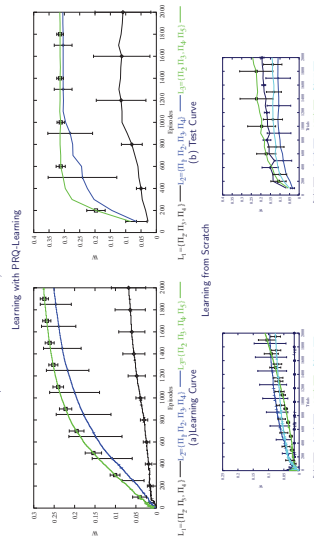
- For  $k = 1$  to  $K$  do
  - Choose an action policy,  $\Pi_k$ , assigning to each policy the probability of being selected according by the following equation:  $P(\Pi_j) = \frac{e^{-W_j}}{\sum_{p=0}^n e^{-W_p}}$  where  $W_j$  is set to  $W_\Omega$
  - Execute the learning episode  $k$ 
    - \* If  $\Pi_k = \Pi_\Omega$ , execute a Q-learning episode following a fully greedy strategy
    - \* Otherwise, use the  $\pi$ -reuse exploration strategy
    - \* Call  $\pi$ -reuse( $\Pi_k, 1, H, \psi, v$ )
    - \* In any case, receive the reward obtained in that episode, say  $R$ , and the updated Q function,  $Q_\Omega(s, a)$
  - Set  $W_k = \frac{W_k + R}{2}$
  - Set  $L_k = L_k + 1$
  - Set  $\tau = \tau + \Delta r$
- Return the policy derived from  $Q_\Omega(s, a)$

19

Fernández and Veloso, December 2005.

## Results

- Parameters: the same as before, with  $\tau = 0$  and  $\Delta r = 0.05$

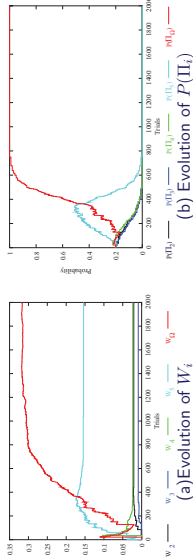


21

Fernández and Veloso, December 2005.

## Evolution of the Estimated Reuse Gains

- PRQ-Learning reusing  $L_3 = \{\Pi_2, \Pi_3, \Pi_4, \Pi_5\}$



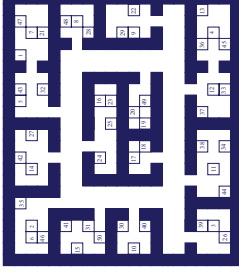
## PLPR Algorithm

### PLPR Algorithm

- Given:
  1. A Policy Library,  $L$ , composed of  $n$  policies,  $\{\Pi_1, \dots, \Pi_n\}$
  2. A new task  $\Omega$  we want to solve
  3. A  $\delta$  parameter
- Execute the PRQ-Learning algorithm, using  $L$  as the set of past policies. Receive from this execution  $\Pi_{\Omega}$ ,  $W_{\Omega}$  and  $W_{max}$ , where:
  - \*  $\Pi_{\Omega}$  is the learned policy
  - \*  $W_{\Omega}$  is the average gain obtained when the policy  $\Pi_{\Omega}$  was followed
  - \*  $W_{max} = \max_i W_i$ , for  $i = 1, \dots, n$
- Update PL using the following equation:

$$L = \begin{cases} L \cup \{\Pi_{\Omega}\} & \text{if } W_{max} < \delta W_{\Omega} \\ L & \text{otherwise} \end{cases} \quad (2)$$

## Tasks



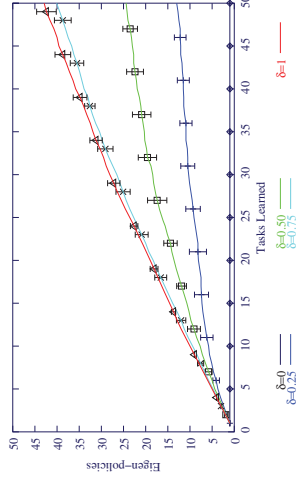
## $\delta$ -similarity

- $\delta$ -similarity between policies:
  - \* Given:
    - \* a policy,  $\Pi$ , that solves a task  $\Omega_i = \langle \mathcal{D}, R_i \rangle$ ,
    - \* a new task  $\Omega = \langle \mathcal{D}, R_{\Omega} \rangle$ , and its respective optimal policy,  $\Pi$
    - \*  $W_{\Omega}$ , or the average gain obtained in  $\Omega$  when the policy  $\Pi$  is followed
    - \*  $W_i$ , or the Reuse Gain of  $\Pi_i$  on task  $\Omega$
    - \* A parameter  $\delta$ , such as  $0 \leq \delta \leq 1$
  - \* Then  $\Pi$  is  $\delta$ -similar to  $\Pi_i$  if  $W_i > \delta W_{\Omega}$
- $\delta$ -similarity with respect to a policy library
  - \* Given:
    - \* a Policy Library,  $L = \{\Pi_1, \dots, \Pi_n\}$  in a domain  $\mathcal{D}$
    - \* a new task  $\Omega = \langle \mathcal{D}, R_{\Omega} \rangle$ , and its respective optimal policy,  $\Pi$
    - \* A parameter  $\delta$ , such as  $0 \leq \delta \leq 1$
  - \* Then,  $\Pi$  is  $\delta$ -similar with respect to  $L$  iff  $\exists \Pi_i \in L$  such as  $\Pi$  is  $\delta$ -similar to  $\Pi_i$ .

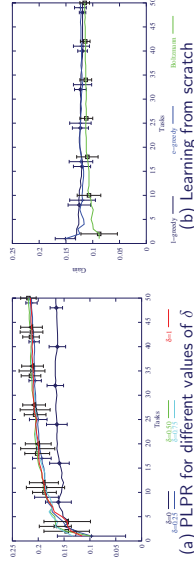
## Eigen Analysis of Policy Reuse

- A Policy Library,  $L = \{\Pi_1, \dots, \Pi_n\}$  in a domain  $\mathcal{D}$  is a  $\delta$ -Basis-Library of the domain  $\mathcal{D}$  iff:
  1.  $\forall \Pi_i \in L$ , such as  $\Pi_i$  is  $\delta$ -similar with respect to  $L - \Pi_i$
  2. every policy  $\Pi$  in the space of all the possible policies in  $\mathcal{D}$  is  $\delta$ -similar with respect to  $L$ .
- Given a  $\delta$ -Basis-Library,  $L = \{\Pi_1, \dots, \Pi_n\}$  in a domain  $\mathcal{D}$ , a new task  $\Omega = \langle \mathcal{D}, R_{\Omega} \rangle$ , each policy  $\Pi \in L$  is called a  $\delta$ -Eigen-Policy of the domain  $\mathcal{D}$  in  $L$
- **Theorem.** The PLPR algorithm builds a  $\delta$ -Basis-Library if:
  1. the reuse gains are accurately computed
  2. the similarity function is symmetric
  3. the PLPR algorithm is executed infinite times over random tasks.

## Number of Eigen-Policies Obtained

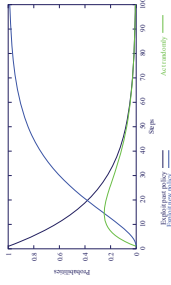


### Building a Policy Library Gain Obtained in the Lifelong Term



### Conclusions Evolution of the Selection Probabilities in each Episode

- Heuristic:
  - \* Initialize  $\psi = 1$
  - \* Decay  $\psi$  with a factor of  $v$  in each step of the episode
- Example:
  - \* Parameters:  $H = 100$ ,  $\psi = 1$ ,  $v = 0.95$ ,  $\epsilon = 1 - \psi$



### Building a Policy Library Eigen-Policies Obtained ( $\delta = 0.25$ )

