Exploration Strategies for Reusing Past Policies

Abstract

The balance between exploring new actions and states, or exploiting the knowledge acquired while learning has been widely studied in Reinforcement Learning. There is also a clear interest in how past policies that solve different tasks may help to solve a new one, and it also requires a balance between explore, exploite past policies or to exploit the current one. In this paper, we show that reusing a past policy can help in the exploration process of the learning of a new policy. We define different exploration strategies which improve the learning efficiency, measured in number of trials required to correctly solve a new task. The experiments demonstrate the usefulness of strategies with an intelligent bias to reuse past similar policies.

1. Introduction

Exploration and exploitation strategies play an important role in Reinforcement Learning (RL). Balancing exploration and exploitation is typically exemplified with the multi-armed bandit problem (Robbins, 1952), and tries to define whether to explore new or exploit the knowledge already acquired (Auer et al., 1995). The optimal strategy is one that minimizes the exploration of the domain while maximizes the quality of the policy learned. However, both optimization problems are opposite.

Reusing sub-policies which were learned for a different but related task can also be used to minimize the experience required to solve a new task. For instance a subproblem of an MDP can be defined as a new MDP where the state space is a subset of the original one. Then, the original MDP can be solved reusing policies learned for different subproblems (Bowling & Veloso, 1999). Intra-Option Learning (Sutton et al., 1998) and TTrees (Uther, 2002) also reuse macro-actions to learn new action policies, in both cases, in Semi-Markov Decision Processes.

Reusing a past policy is appealing because it may bias a new learning process. However, it is still a challenge, given that biasing the learning inherently complicate the exploration and exploitation balance. That is because in addition to the classical balance between exploring new states or exploiting the current policy, it adds a third factor of exploiting the past policy. However, this balance has been successfully found in other problems like path planning, where reusing waypoints used in past plans has demonstrated to be useful to solve new planning problems (Bruce & Veloso, 2002).

In this work, we define and analyze different strategies to reuse a past policy. The reuse is performed though biasing the exploratory process of the new policy. These strategies define when to explore, when to follow the past policy, and when to exploit the knowledge acquired in the current learning process. The scope of this research is direct reinforcement learning, and the Q-Learning (Watkins, 1989) algorithm will be used in all the experiments, although the strategies are general, and they only require off-policy algorithms (i.e. algorithms able to learn an optimal policy while following a different one).

The paper is organized as follow. Next section summarizes the related work, focusing on exploration strategies and in policy reuse methods. Section 3 formalizes the concepts of tasks and domains, and illustrate them with an example. Section 4 describes the new exploration strategies for biasing new learning processes with a past policy. Section 5 describes the experiments performed, while Section 6 describes the main conclusions and further research.

2. Related Work

In the literature, different kinds of exploration strategies can be found. A random strategy always selects randomly the action to execute, without to use the knowledge is being acquired. The ϵ-greedy strategy selects with a probability of ϵ the best action suggested by the Q function learned up to that moment, and it selects a random action with probability of $(1 - \epsilon)$. Boltzmann strategy tries to rank the actions, provid-

---

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ing the actions with a higher value of Q with a higher probability, as defined in equation 1, for a set of n actions. This equation also includes a τ parameter that moves the strategy from random, when $\tau = 0$, to greedy while τ grows (Sutton & Barto, 1998).

$$P(a_j) = \frac{e^{\tau Q(s,a_j)}}{\sum_{p=1}^{n} e^{\tau Q(s,a_p)}} \tag{1}$$

These strategies may also include small modifications, as the optimistic initialization of the Q values, which increases exploration in earlier steps of the learning.

Directed exploration strategies memorize exploration-specific knowledge that is used for guiding the exploration search(Thrun, 1992). These strategies are based in heuristics that bias the learning so unexplored states tend to have a higher probability of being explored that recently visited ones. However, they require a model of the domain (the state transition function) to execute the heuristics.

Task clustering can also be used to organize and reuse different tasks (Thrun & O'Sullivan, 1995). Other methods try to learn environment independent knowledge so the learned knowledge can be used for similar tasks in different scenarios (Thrun & Mitchell, 1995). Reusing the Q function that represents a policy learned for a task can also be useful if it is similar to the new one (Carroll & Peterson, 2002). However, it requires the Q function is available, and not only the policy.

However, most of the previous examples are focused only on exploration or in policy reuse, but they do not handle the case of biasing exploration with a past policy.

3. Domains and Tasks

We describe the concept of domain and task, and we stablish the scope of this research.

3.1. Definitions

A Markov Decission Proccess (Puterman, 1994) is represented with a tuple $< \mathcal{S}, \mathcal{A}, \delta, \mathcal{R} >$, where $\mathcal{S}$ is the set of all the possible states, $\mathcal{A}$ is the set of all the possible actions, δ is an unknown stochastic state transition function, $\delta : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \Re$, and $\mathcal{R}$ is an unknown stochastic reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$. However, in this work, we focus in RL domains where different tasks can be solved. Different tasks are given different reward functions, but the other concepts, S, A and δ and stay constant for all the tasks. Thus, we break the MDP concept in two: domain and task.

We characterize a domain, $\mathcal{D}$, as a tuple $< \mathcal{S}, \mathcal{A}, \delta >$, We define a task, Ω, as a tuple $< \mathcal{D}, \mathcal{R}_\Omega >$, where $\mathcal{D}$ is a domain as defined before, and $\mathcal{R}_\Omega$ is the stochastic and unknown reward function. Thus, different tasks may have different reward functions.

We define an action policy, Π, as a function $\Pi : \mathcal{S} \rightarrow \mathcal{A}$. If the action policy was created to solve a defined task, Ω, the action policy is called Π_Ω. We can assume that we are solving a task with absorbing goal states. Thus, if s_i is a goal state, $\delta(s_i, a, s_i) = 1$, $\delta(s_i, a, s_j) = 0$ for $s_i \neq s_j$, and $\mathcal{R}(s_i, a) = 0$, for all $a \in \mathcal{A}$. Each trial finishes when a goal state is reached or when a maximum number of steps, say H, is achieved. The value to minimize is the average number of steps executed to achieve the goal in each trial, say T, as defined in equation 2:

$$T = \frac{1}{K} \sum_{k=0}^{K} t_k \tag{2}$$

where t_k defines the number of steps performed to arrive to the goal in the trial k. So, we are following a cost-to-go approximation (Bertsekas & Tsitsiklis, 1996), which, instead of meausuring the average reward, measures the cost to arrive the goal.[1]. (Manuela, I think that the last sentence, as well as the footnote are not necessary, and could be deleted. I guess that instead of clarify, they may confuss. What do you think??)

Lastly, an optimal action policy for solving the task Ω is called Π^*_Ω. To be optimal means that $T_\Omega^{\Pi^*} \leq T_\Omega^{\Pi}$, for all policy Π in the space of all possible policies.

3.2. Scope

The scope of this work is the following:

- 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 $\{\Pi^*_{\Omega_1}, \ldots, \Pi^*_{\Omega_n}\}$
- Let's assume that there is a supervisor who, given Ω, tells us which is the most similar task, Ω_i to Ω

In this work we assume two issues. Firstly, that it exists a supervisor who provides a policy that solves

[1]This is done to allow the trials which start far from the goal to make also a strong apportation to the quality measure of the policy. Average reward received in each trial avoids that when delayed reward is used, given that the γ^t becomes very close to 0 when t is high and typical values of γ (as 0.9) are used

a task similar to the one that we are trying to solve. Differences between both tasks are derived from differences in their respective reward functions, which are unknown. Then, similar tasks share the policy for most of the state space, and only a region of the state space have a different policy. These concepts will be formalized in Section 3, although we will not discuss in this paper how task similarity can be measured, nor how that supervisor can be implemented. That discussion can be found in (under submission, 2005).

Second, we trust the supervisor's suggestion, i.e. we do not study the effects of trying to reuse a policy that solve a task completely different to the one we are trying to solve now. Deciding whether to trust or not the supervisor also relies on deciding if the past task is, indeed, similar or not to the one we are trying to solve now. As introduced above, that discussion is out of the goals of this paper.

Given previous elements, can we speed up the learning of Π^*_Ω? To do that, exploration strategies able to bias the new learning process with the one received by the supervior are required. The next section describes several different strategies to incorporate this bias.

4. Exploratory Strategies for Policy Reuse

In this section, several exploration strategies are defined. We denote the old policy with Π^{old}, and the one we are currently learning with Π. We assume that we are using a direct RL method to learn the action policy, so we are learning its related Q function. Any algorithm can be used to learn the Q function, with the only requirements that it can learn off-policy, i.e. it can learn a policy while executing a different one, as Q-Learning does (Watkins, 1989).

The exploration strategies are divided in two different groups, depending on if they use only the past policy (Π^{old}), or if they also use the new one (Π).

4.1. Exploration strategy ψ-reuse

The goal of this policy is to balance random exploration and exploiting the old policy. There are two approaches for this strategy. The first one uses the past policy, Π^{old}, with a probability of ψ. It selects an action randomly with a probability of $1-\psi$. Thus, this policy is similar to ϵ-greedy, but instead of exploiting the current policy, Π, it exploits the old one, Π^{old}. Then, balance between exploring and exploiting the past policy is constant in each trial.

The second approach uses the past policy with a probability of ψ, and acts randomly with a probability of $1-\psi$, as well as the previous approach does. However, in each step of the trial, ψ decays a factor of υ, so in each step, ψ_{h+1} is set to $\upsilon\psi_h$. The idea of this strategy is that it exploits the past policy in the initial steps of each trial, but gradually explores randomly. Both approaches are merged in Table 4.1, where the ψ-policy exploration strategy is described. The first approach is included also in this algorithm, setting $\upsilon = 1$.

ψ-reuse Strategy (K, H, ψ, υ).
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, $a = \Pi^{old}(s)$
With a probability of $1-\psi_h$, $a = rand(\mathcal{A})$
Receive current state s', and reward, r
Update $Q^\Pi(s,a)$
Set $\psi_{h+1} \leftarrow \psi_h \upsilon$
Set $s \leftarrow s'$

Table 1. ψ-reuse Strategy.

The problem of this strategy is that it never exploits the knowledge that it is acquiring. Next strategy solves this problem introducing also the new policy in the action selection process.

4.2. Exploration Strategy ψ-combined-reuse

The goal of this policy is to balance random exploration, exploiting the old policy, and exploiting the policy is being learned. The new strategy is similar to to $\psi - reuse$, given that it also follows the past policy with a probability of ψ. However, with a probability of $1-\psi$, it exploits the new policy. Obviously, random exploration is always required, so when exploiting the new policy, it follows an $\epsilon - greedy$ strategy, as it is defined in Table 4.2.

Thus, there are three probabilities involved: the probability of exploiting the past policy, the probability of using current policy, or the probability of acting randomly. These probabilities are shown in Figure 1, for input values of $H = 100$, $\psi = 1$ and $\upsilon = 0.95$. In this case the ϵ parameter is set in each step to $1 - \psi_h$.

The figure shows that in the initial steps of each trial, past policy will be exploited. As the number of steps increases, exploration also increases, while in the ending steps of the trial, the new policy will be exploited. The transition from exploiting past policy and exploiting the new one depends on the υ parameter. If this

ψ-combined-reuse Strategy (K, H, ψ, υ).

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, $a = \Pi^{old}(s)$
    With a probability of $1 - \psi_h$,
      $a = \epsilon - greedy(\Pi(s))$
    Receive current state s', and reward, r
    Update $Q^{\Pi}(s, a)$
    Set $\psi_{h+1} \leftarrow \psi_h \upsilon$
    Set $s \leftarrow s'$

Table 2. ψ-combined-reuse Strategy.

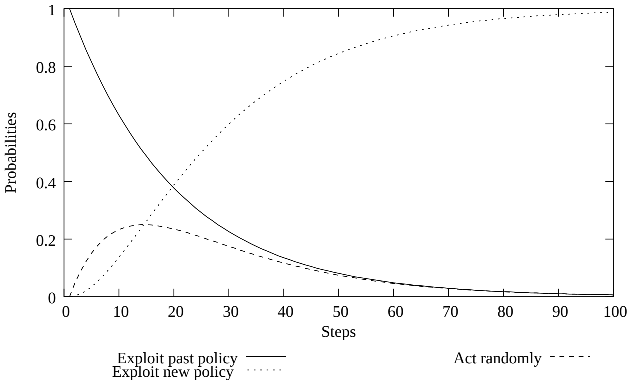


Figure 1. Evolution of the probabilities of exploring and exploiting in a trial for the ψ-combined-reuse exploration strategy.

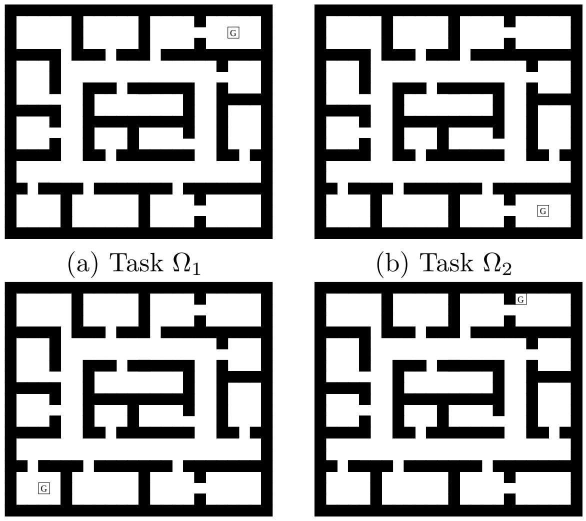
parameter is low, the transition occurs in the initial steps, while if it is high, the transition is delayed.

5. Experiments

In this section we describe the experiments performed to demonstrate the usefulness of the exploration strategies defined above. But firstly, we describe the domain used.

5.1. Tasks in a Robot Navigation Domain

This domain consists of a robot moving inside of an office area, as shown in Figure 2, similar to the one used in other RL works (Fernández & Borrajo, 2002; Thrun & Schwartz, 1995). This area is represented by walls, free positions and goal areas, all of them of size 1×1. The whole domain is $N \times M$ (24×21 in this case). The possible actions that the robot can execute are "go North", "go East", "go South" and "go West", all of size one. The robot knows its location in the space through continuous coordinates $(x + n_x, y + n_y)$ provided by some localization system, where x and y are the real coordinates, and n_x and n_y correspond with a noise in the perception, and are defined as random variables following a uniform distribution in the range $(-0.20, 0.20)$. In this work, we assume that we have the optimal uniform discretization of the state space (which consists on 24×21 regions). Furthermore, the robot has an obstacle avoidance system that blocks the execution of actions that would crush it into a wall. The goal in this domain is to reach the area marked with 'G'. When the robot achieves it, it is considered a successful trial, and it receives a reward of 1. Otherwise, it receives a reward of 0.



(a) Task Ω_1 (b) Task Ω_2

(a) Task Ω_3 (b) Task Ω

Figure 2. Four different tasks in the robot navigation domain.

Figure 2 shows the same domain, but four different task, given that the goal states, and hence, the reward functions, are different in each of them. Biasing the learning of Π using Π_1 (policy that solves Ω_1) seems to be useful given that, as shown in figure 3, both policies could be equal for a big number of states. However, what states may share the same policy and what areas may not is completely unknown a priory, given that the reward function, as well as the state transition function, are unknown.

5.2. Description of the Learning Curves

In the following subsections, we will describe the experimental results of applying different exploration strategies. For each of these strategies (and parameter settings), we will present two results showing two different curves, the learning curve, and the test curve.

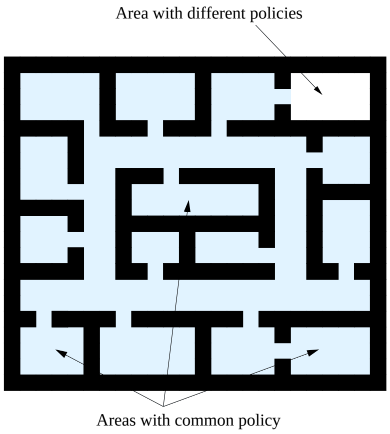


Figure 3. Common policy for the two tasks described in Figure 2

The learning curve of each strategy describes the performance of such strategy in the learning process. Learning has been performed using the Q-Learning algorithm, for fixed parameters of $\gamma = 0.9$ and $\alpha = 0.2$. [2] Learning with each strategy is performed once. This learning consists on executing $K = 2000$ trials. Each trial consists on following the defined strategy until the goal is achieved or until the maximum number of steps, $H = 100$, is executed. In the x axis of the figure, the trial number is shown. The y axis represents the average number of steps per trial executed since the learning started. Thus, a value of 50 for the trial 200 means that the average number of steps executed in the 200 first trials has been 50. If in one of those trials, the goal was not achieved, it is penalized with a value of 100.

The test curve represents the evolution of the performance of the policy while it is being learned. Each 100 trials of the learning process, the Q function learned up to that moment is stored. Thus, after the learning process, we can test all those policies. Each test consists on 1000 trials where the robot follows a completely greedy strategy. Thus, the x axis shows the learning trial in which that policy was generated, and the y axis show the result of the test, measured as the average number of steps executed to achieve the goal in the 1000 test trials.

[2] A value of $\gamma = 0.9$ has demonstrated to solve this task efficiently. α is typically set to a higher value at the beginning of the learning, and is decreased progressively to ensure convergence. We have fixed the value of α to avoid that different reduction factors may favor to different strategies. A fixed value of α does not ensure the convergence of the learning process (Watkins & Dayan, 1992), but avoids interferences in the analysis

5.3. Learning from Scratch

Firstly, the learning and test processes have been executed following different exploratory strategies that does not use the past policy. Specifically, we have used three different strategies. The first one is a random strategy. The second one is Boltzmann for a fixed $\tau = 1000$, that stays constant during the whole learning. This is, hence, a very greedy strategy. Lastly, Boltzmann strategy has been also followed, but initializing $\tau = 0$, and increasing it in 5 in each learning trial. Figure 4 shows the learning and test curves for all them.

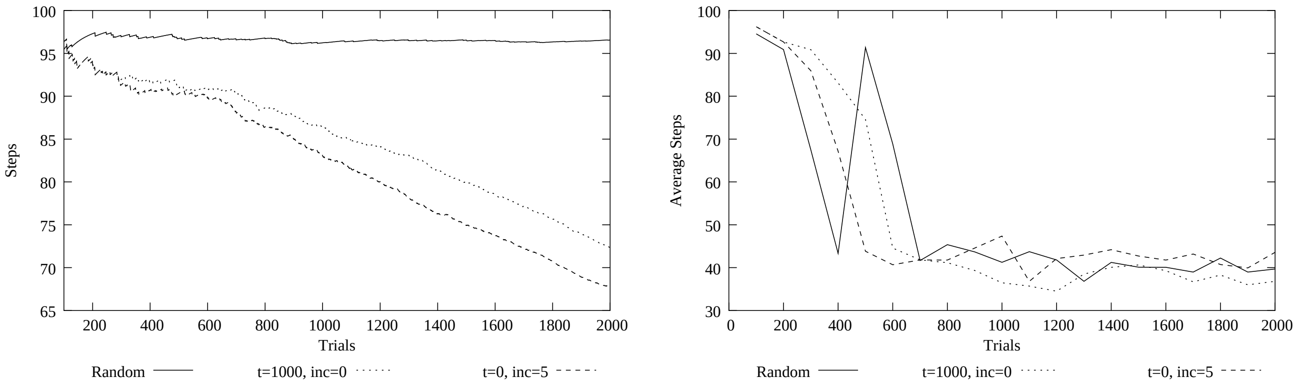
We focus first in the learning curve shown in Figure 4(a). We see that when acting randomly, the average number of steps in learning is close to 100. To reduce this value, Boltzmann strategy is required. In this case, the most greedy strategy (fixed value of $\tau = 1000$) seems to behave better in learning obtaining values below to 70 after the 2000 trials. With this parameters, the behavior is also better in test. For the random strategy, cost decreases in test very fast, showing that it behaves very good in test.

Lastly, notice that the quality of the policy learned is very sensible to small changes in some crucial regions, as the regions close to door of the office where the goal is. A change in such region makes that the robot does not arrive to the goal in most of the trials, increasing extremely the average number of steps to goal (given that the trials which not ends in goal introduce a cost of 100 steps). That generates out-layers in the learning curves, as the one shown in the test curve of random strategy for the iteration 500. These out-layers will appear also in further experimental results. Incorrect estimations of the right policy are due to the stochasticity of the domain, together with the constant value of α.

5.4. Reusing the Past Policy Following ψ-reuse

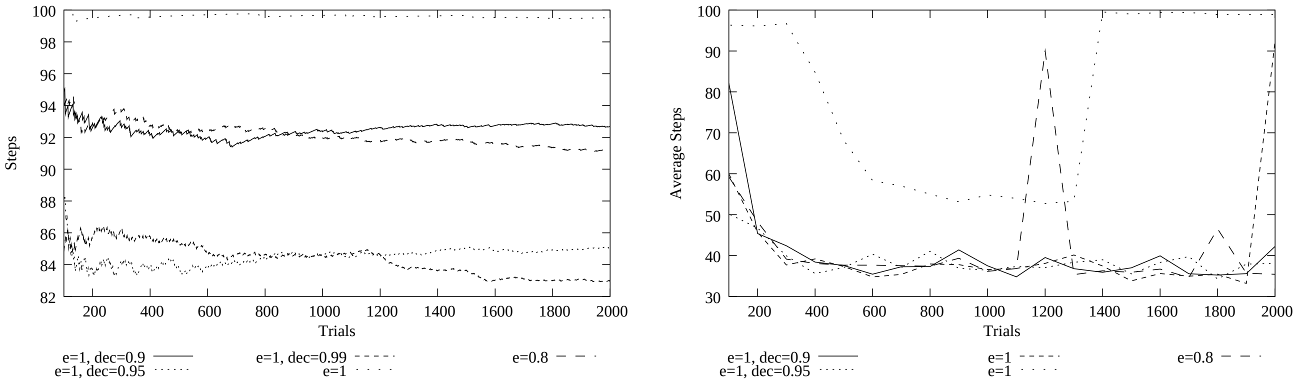
Figure 5 shows the learning and test curves when performing the exploration using the ψ-reuse strategy. The experiment has been performed for different combinations of the ψ and υ parameters.

In the learning curve it is shown that there is not improvement in learning, given that this policy never exploit the current policy. That makes that the average number of steps in the learning process stays constant for all them. However, there are strong differences depending on the parameters used. For instance, the exploration process that is only biased by the past policy correspond with the set of parameters $\psi = 1$ and the decrement rate of $\tau = 1$. The learning curve for



(a) Learning Curve (b) Test Curve

Figure 4. Learning and test evolution when learning from scratch



(a) Learning Curve (b) Test Curve

Figure 5. Learning and test evolution when following ψ-reuse exploration strategy.

this strategy shows that the average number of states is close to 100, which means that the goal is not achieved in most of the learning trials, given that it is trying to achieve the goal of the past policy in a completely greedy way. That produces that a correct policy is not learned, as is demonstrated in the test curve. This result can be improved by allowing some exploration. A way to do it is setting τ to 0.8, which means that the 20% of the steps, a random action is selected. This decreases the average number of steps per trial to 92 after the 2000 trials. That improvement in learning is also reflected in the test.

Lastly, the best results are obtaining when initializing ψ to 1, with decreasing rates of $\tau = 0.95$ or 0.99. With both set of parameters, the average number of steps is around 85 in learning. Test curves for these sets of parameters show that learning is very fast and in only 100 iterations, the cost of arriving the goal is under 60. In both cases, after 400 trials the value is below 40 in test.

However, if the discount factor is strong ($\tau = 0.9$) the results in learning are not so good, given that the $\psi 4$ parameter becomes close to 0 very fast, and hence, the past policy is almost not used in the exploration. However, it also shows good results in test.

5.5. Reusing the Past Policy Following ψ-combined-reuse

Figure 6 shows the results of using the ψ-combined-reuse exploration strategy. We have executed the strategy for two different sets of parameters. In both cases, ψ has been set to 1, but in the first case, the discount factor υ has been set to 0.95 and 0.9 respectively.

Opposite to the results of exploring with ψ-reuse, the learning curve decreases while learning is performed, given that in this exploration strategy, the policy is being learned is also exploited. In this case, the learning curve for $\upsilon = 0.95$ shows that in training, a low number of steps is achieved much faster, and in only 200 trials, it is under 50. Test curve shows that the policy learned is also better when learning is performed with

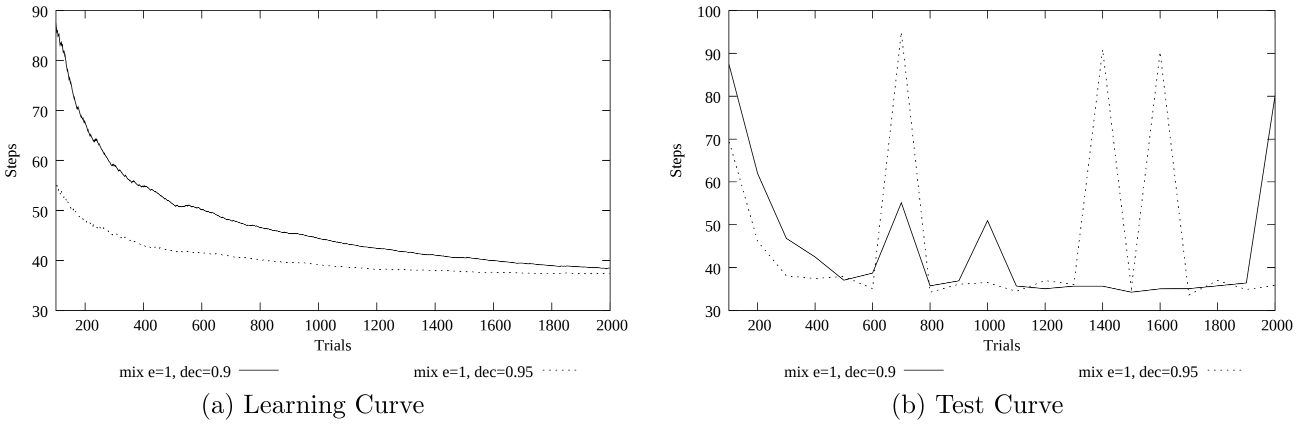


Figure 6. Learning and test evolution when following ψ-combined-reuse exploration strategy.

$\upsilon = 0.95$, and the policy learned after only 300 steps produces a cost under 40.

However, the test curve obtained with this exploration strategy shows that the learned policies are unstable, and that a lot of out-layers in the cost are generated.

5.6. Experiments Summary

Figure 7 summarizes the results obtained. It shows the best result obtained with each exploration strategy: (i) Boltzmann with an initial $\tau = 0$, which is increased in each step a value of 5; (ii) ψ-reuse with $\psi = 1$ and $\upsilon = 0.95$; and (iii) ψ-combined-reuse with $\psi = 1$ and $\upsilon = 0.95$.

The figure shows that the best results in training are obtained by ψ-combined-reuse strategy. The improvement of this strategy is very great when compared to the others. That makes this exploration strategy very useful for new learning problems, where the cost of gathering experience is very high. However, this strategy generates very sensitive policies, as the out-layers produced in test demonstrate. However, this could be solved with a more accurate selection and evolution of the α parameter. In any case, it does not influence in a real execution process, where the curve followed is the learning curve.

6. Conclusions and Further Research

In this work, the reuse of a past policy to speed up the learning of a new task has been studied. Two exploration strategies that allow to bias the exploration with the past policy have been presented, ψ-reuse and ψ-combined-reuse. They have demonstrated its usefulness to improve the learning of a new action policy over exploration strategies that does not reuse any knowledge.

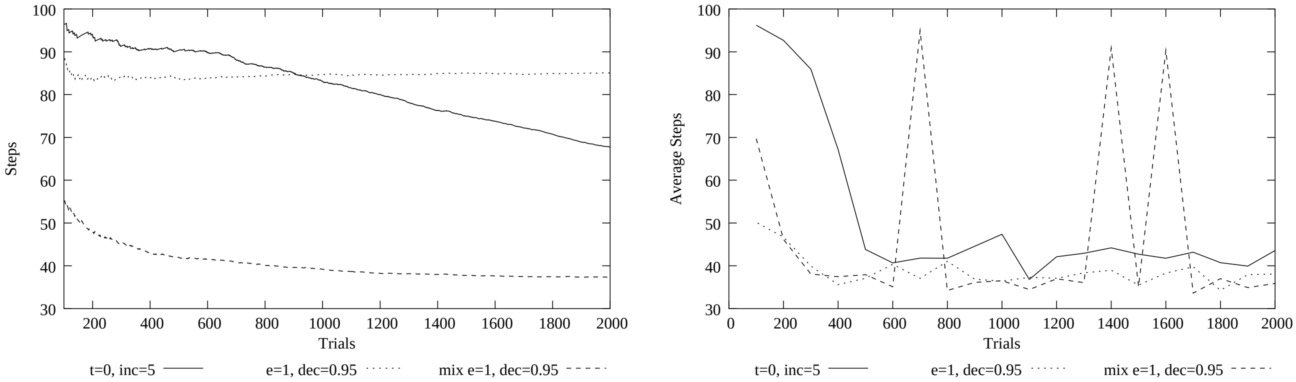
Obviously, very different strategies can be derived from the ones presented in this work. For instance, in ψ-combined-reuse, the balance among exploring, exploiting the past policy and exploiting the new one is static in all the learning process. Heuristics that exploit past policies in the beginning of the learning, and that progressively exploits more the current one could be applied. To do this, the initial value of ψ could be reduced in each trial, or the value of υ could also be reduced in each trial, increasing the exploitation of the current policy.

However, policy reuse, as it has been presented in this work, requires of a supervisor who provides a useful policy previously learned. This work assumes that this supervisor exists and that it provides a policy corresponding with a task that is "similar" to the one we are trying to solve. We have not studied in this work how this supervisor could be implemented, nor how the similarity metric could be. However, we believe that defining good exploration strategies able to incorporate past knowledge in a new learning process is a required step to the problem of reusing past policies when the policy to reuse is not known in advance.

Thus, future work will focus on how these exploration strategies may help to speed up new learning processes reusing past policies, even when it is unknown a priori which of those policies are useful, or even whether it exists or not.

References

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. *36th Annual Symposium on Foundations of Computer Science* (pp. 322–331).



(a) Learning Curve (b) Test Curve

Figure 7. Learning and test evolution following different exploration strategies.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Bellmon, Massachusetts: Athena Scientific.

Bowling, M., & Veloso, M. (1999). Bounding the sub-optimality of reusing subproblems. *Proceedings of IJCAI-99*.

Bruce, J., & Veloso, M. (2002). Real-time randomized path planning for robot navigation. *Proceedings of IROS-2002*. Switzerland. An earlier version of this paper appears in the Proceedings of the RoboCup-2002 Symposium.

Carroll, J., & Peterson, T. (2002). Fixed vs. dynamic sub-transfer in reinforcement learning. *Proceedings of the International Conference on Machine Learning and Applications*.

Fernández, F., & Borrajo, D. (2002). On determinism handling while learning reduced state space representations. *Proceedings of the European Conference on Artificial Intelligence (ECAI 2002)*. Lyon (France).

Puterman, M. L. (1994). *Markov decision processes - discrete stochastic dynamic programming*. New York, NY.: John Wiley & Sons, Inc.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, *55*, 527–535.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, Massachusetts: MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1998). Intra-option learning about temporally abstract actions. *Proceedings of the Internacional Conference on Machine Learning (ICML'98)*.

Thrun, S. (1992). *Efficient exploration in reinforcement learning* (Technical Report C,I-CS-92-102). Carnegie Mellon University.

Thrun, S., & Mitchell, T. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*, 25–46.

Thrun, S., & O'Sullivan, J. (1995). *Clustering learning tasks and the selective cross-task transfer of knowledge* (Technical Report CMU-CS-95-209). School of Computer Science, Carnegie Mellon University.

Thrun, S., & Schwartz, A. (1995). Finding structure in reinforcement learning. *Advances in Neural Information Processing Systems 7*. MIT Press.

under submission, A. (2005). Learning by probabilistic reuse of past policies. *Proceedings of the International Joint Confereneces on Artificial Intelligence (IJCAI'05)*.

Uther, W. T. B. (2002). *Tree based hierarchical reinforcement learning*. Doctoral dissertation, Carnegie Mellon University.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Doctoral dissertation, King's College, Cambridge, UK.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-Learning. *Machine Learning*, *8*, 279–292.