Reusing Risk-Aware Stochastic Abstract Policies in Robotic Navigation Learning*

Valdinei Freire da Silva[1], Marcelo Li Koga[2],
Fábio Gagliardi Cozman[2], and Anna Helena Reali Costa[2]

[1] Universidade de São Paulo – Escola de Artes, Ciências e Humanidades – São Paulo, Brazil.
[2] Universidade de São Paulo – Escola Politécnica – São Paulo, Brazil.
{valdinei.freire,mlk,fcozman,anna.reali}@usp.br

Abstract. In this paper we improve learning performance of a risk-aware robot facing navigation tasks by employing transfer learning; that is, we use information from a previously solved task to accelerate learning in a new task. To do so, we transfer risk-aware memoryless stochastic abstract policies into a new task. We show how to incorporate risk-awareness into robotic navigation tasks, in particular when tasks are modeled as stochastic shortest path problems. We then show how to use a modified policy iteration algorithm, called AbsProb-PI, to obtain risk-neutral and risk-prone memoryless stochastic abstract policies. Finally, we propose a method that combines abstract policies, and show how to use the combined policy in a new navigation task. Experiments validate our proposals and show that one can find effective abstract policies that can improve robot behavior in navigation problems.

Keywords: Risk-Awareness, Memoryless Stochastic Abstract Policies, Transfer Learning

1 Introduction

Consider an agent that displays risk-awareness in the sense that she has preferences amongst risk-prone, risk-neutral and risk-averse attitudes. This agent may face a series of tasks; for instance a robot may face navigation problems in a variety of rooms, each one in a different day. It is reasonable to suppose that insights obtained in the solution of the first task will be useful in the solution of the second task, and so on. Hence the agent may be interested in learning not only the solution of the first task, but also an *abstract* description of the solution that may be used when solving the second task, and so on. How can this risk-aware agent find a suitable abstract policy? How can the agent reuse and transfer into new problems this abstract policy, and how useful can it be?

* This research was partly sponsored by FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo (Procs. 11/19280-8, 12/02190-9, and 12/19627-0) and CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico (Procs. 311058/2011-6 and 305395/2010-6).

Substantial previous work can be found on transfer learning methods. One must decide *what* to transfer: value functions [12,19], features extracted from the value functions [1,8], heuristics and policies [3,5]. We focus on policy transfer; all transferred information is encoded by policies. An advantage of policy-based transfer is that it requires only a mapping between the states/actions of the *source* task and the states/actions of the *target* task; this amounts to less information than required by methods that transfer value functions [5]. Policy-search is in fact quite appropriate when coupled with abstraction [9]. Our approach is to transfer learning by exploiting abstract policies encoded through *relational* representations that compactly capture domains in terms of relations and objects [14]. Each abstract state aggregates a set of concrete states; given a single abstract state, one cannot know which concrete state obtains, but each concrete state is mapped to a unique abstract state. We consider abstract policies that are *memoryless* and *stochastic*. We use the AbsProb-PI algorithm [17] to construct such abstract policies in such a way that abstraction captures the main features of the source task. A memoryless policy chooses actions only according to the current observation of the system. No matter how smart the agent may be in constructing the abstract policy from the source problems, a direct application of an abstract policy in a new problem will almost certainly generate sub-optimal behavior, so one must always mix actions prescribed by an abstract policy with actions learned in the new, concrete task.

Our main contribution in this paper is to incorporate risk-awareness into the abstract policy built from a source task, so that the agent can modulate her prior experience by her preferences over risk, when learning features of a new target task. Introducing explicit preferences over risk requires us to measure subjective attitudes towards risk and to evaluate policies as risky or conservative [6,15]. A decision maker can have three distinct attitudes towards risk: averse, neutral or prone [6]. Relatively few activity in AI and Robotics addresses risk-awareness in policy construction, despite the practical importance of this issue, possibly due the difficulty in optimizing risk-aware measures. Liu and Koenig [11] consider probabilistic planning with nonlinear utility functions. Delage and Mannor [4] consider a relaxed version of minimax by defining probabilistic constraints on minimal performance. Mannor and Tsitsiklis [13] consider a trade-off between expected and variance of accumulated rewards. In all of these approaches, optimal policies are non-stationary in general.

We focus on robotic navigation problems modeled as Stochastic Shortest Path (SSP) problems.

We show here that an SSP problem, modeled as an MDP using the infinite-horizon discounted-reward criterion, can incorporate risk-awareness by varying the discount factor [15]. We then apply the AbsProb-PI algorithm [17] in the construction of memoryless stochastic abstract policies showing different attitudes towards risk, as a consequence of different discount factors. We also propose a strategy that balances between risk-neutral and risk-prone attitudes when reusing abstract policies in a new target task. Our experiments show that at the beginning of a new task, better results are obtained by using a risk-prone

attitude, and as the execution of the task progresses, it is better to adopt a risk-neutral attitude. Both risk-prone and risk-neutral attitudes are dominated by the *non-stationary* policy that combines both attitudes in the new target task.

The remainder of this paper is structured as follows. Section 2 presents our proposals for tuning the discount factor in order to incorporate risk-awareness. Section 3 explains how to use relational representations in MDPs, describes the AbsProb-PI algorithm and presents our approach to combine different risk-aware policies. We present experimental results and analysis in the robotic navigation domain in Section 4. Finally, in Section 5 we draw our final conclusions.

2 Modeling Risk Sensitivity in SSPs with Negative Constant Rewards

In this section we formalize Markov Decision Processes (MDPs) and the Stochastic Shortest Path (SSP) problem we use when modeling the robotic navigation task. We will show that risk-awareness in this setting can be equated with the choice of the discount factor used in evaluating the performance of an agent in an MDP.

2.1 Markov Decision Process

For our purposes an MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T_a(\cdot), \beta, r(\cdot) \rangle$, where $\mathcal{S}$ is a finite set of process states s; $\mathcal{A}$ is a finite set of possible actions a; $T_a : \mathcal{S} \times \mathcal{S} \to [0, 1]$ is a conditional probability distribution over $\mathcal{S}$ given each state in $\mathcal{S}$; $\beta : \mathcal{S} \to [0, 1]$ is an initial probability distribution; and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function [16]. An MDP starts at a state $s_0 \in \mathcal{S}$, selected with probability $\beta(s_0)$. If the process is in state i at time t and action a is applied, then the next state is j with probability $T_a(i, j)$ and a reward $r(i, a)$ is received. In an MDP, a *stationary stochastic policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ specifies the probability $\pi(s, a)$ of executing action a in state s. We denote by $\mathcal{A}^s$ the set of allowable actions in state s; that is the support of $\pi(s, \cdot)$.

In this paper we wish to consider risk-aware decision profiles; to do so, we introduce a utility function that encodes preferences over the value of histories. We denote by $u(R(h))$ the utility of a history. The quality of a policy is measured by its expected utility:

$$E[u(R(h))] = \sum_h P(h|\pi) u(R(h)). \tag{1}$$

Clearly if $u(x) = \lambda x$ for some $\lambda > 0$, we return to the usual $E[R(h)] = \sum_h P(h|\pi) R(h)$. If instead $u(x) = e^{-\lambda x}$ (exponential utility), then:

$$E[u(R(h))] = \sum_h P(h|\pi) e^{-\lambda R(h)}. \tag{2}$$

An optimal policy π^* is a policy that yields the highest expected utility [16].

2.2 Stochastic Shortest Path Problem

A special case of MDPs is the Stochastic Shortest Path (SSP) problem with a unique goal state $s_G \in \mathcal{S}$ that is an absorbing state, i.e. if $r(s_G, a) = 0\ \forall a \in \mathcal{A}$ and $T_a(s_G, s_G) = 1\ \forall a \in \mathcal{A}_{s_G}$. Additionally, we assume that at each state except s_G the reward function is negative and constant, and fixed at -1. Suppose the decision maker wishes to minimize the number of steps to reach s_G. In this case each history h is summarized by its number of transitions $|h|$, because

$$R(h) = -(1 + \gamma + \gamma^2 + \cdots + \gamma^{|h|-1}). \tag{3}$$

Focus for a moment on $\gamma < 1$. Then

$$R(h) = -\frac{1 - \gamma^{|h|}}{1 - \gamma} = \alpha e^{\eta |h|} - \alpha, \tag{4}$$

where $\alpha = 1/(1-\gamma)$ and $\eta = \ln \gamma$. Note that $\eta < 0$, so this expression for $R(h)$ holds even if history h is infinitely long (we obtain $R(h) = -\alpha$). In fact in this case $-\alpha \leq R(h) \leq 0$, hence $E[R(h)]$ exists for every policy and is finite, a well-known result.

Additionally, if $\gamma = 1$, then $R(h) = -|h|$; in this case a proof that $E[R(h)]$ is finite depends on further conditions [2]. In particular, say that a stationary policy is *proper* if the probability that it reaches s_G starting from any state s goes to 1 as t goes to infinity: If the SSP is such that there exists a proper policy, and any policy that is not proper yields infinite expected value over histories starting at some state, then a finite $E[R(h)]$ exists for some stationary policy.

In fact, with an additional assumption we can even consider values of γ larger than one. For each fixed policy π, write down the Markov chain that is obtained by fixing transition probabilities at π. Now construct an auxiliary Markov chain M_π by removing all arcs from s_G into itself, and adding arcs from s_G to each one of the other states, associating each arc from s_G to s with $\beta(s)$. We say the SSP problem is *recurrent* if each M_π (for each π) defines a recurrent Markov chain [20]. Recall that a finite Markov chain is recurrent if it is irreducible (any state can reach any state with positive probability); and if a finite Markov chain is recurrent, it is positive recurrent (the expected number of transitions to return to a state is finite). If the SSP problem is positive recurrent, every policy is proper. Moreover, Expression (4) holds for values of γ larger than one, and a finite $E[R(h)]$ exists for some stationary policy.

2.3 Risk-awareness as γ selection

Suppose our SSP problem has a proper policy, and every non-proper policy yields infinite expected value over histories starting at some state.

We can interpret $E[R(h)]$ for our SSP problem as the expected utility of *another* SSP problem, with the same states, actions and transition probabilities, but where $\gamma = 1$, and an associated utility function $u(\cdot)$. That is, the original SSP problem is not risk-aware and runs with some $\gamma < 1$, while the associated

SSP problem is risk-aware and runs with $\gamma = 1$. We can do this because when $\gamma = 1$, we have $R(h) = -|h|$, and we can then take the utility function:

$$u(x) = \alpha e^{-\eta x} - \alpha.$$

This device produces an expression analogue to Expression (4). So, the risk-aware analysis with $\gamma = 1$ and utility function $u(\cdot)$ yields the same results as our original SSP without risk-awareness.

Consequently, we can interpret γ in the original problem as an expression of a risk-aware judgement on the part of the agent [15]. We note that there exist other interpretations of γ; for instance, γ may reflect the fact that the agent focuses more intensely in close rewards; another interpretation is that γ is the probability of surviving one more time step [21].

Obviously, if the original problem already has $\gamma = 1$, then the agent already has a risk-neutral attitude. But if $\gamma < 1$ in the original problem, we can say that the agent has a risk-prone attitude.

Now suppose our SSP is recurrent. In this case the analyzes in the previous paragraph holds even for values of γ larger than one in the original problem. In this case we obtain $\eta > 0$ and the utility function reflects a risk-averse behavior.

3 Risk-Sensitive Abstract Policies

We can now use the proposed interpretation of the discount factor γ to model different attitudes toward risk of an agent aimed at solving an SSP. As stated earlier, SSP is a special case of MDP; so we can use any algorithm that solves MDPs to solve our SSP problem. For each solution, we can tune the agent's attitude towards risk by varying γ . However, our interest is not only in tuning the agent's attitude toward risk, but also in abstracting the solution so that this abstract solution can be reused in other new problems.

3.1 Relational Representations and Abstraction

Let us introduce some definitions first. C is a set of *constants*, P_S is a set of *state predicates* and P_A is a set of *action predicates*. If $\mathtt{t}_1, \ldots, \mathtt{t}_n$ are *terms*, each one being a constant (represented with lower-case letters) or a variable (represented with capital letters), and if $\mathtt{p}/n$ is a predicate symbol, $\mathtt{p} \in P_S$ (or $\mathtt{p} \in P_A$), with arity $n \geq 0$, then $\mathtt{p}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ is an *atom*. If an atom does not contain any variable, it is called a *ground atom*. We call a set of atoms a *conjunction*. The state set $\mathcal{S}$ of a relational MDP is defined as the set of possible ground conjunctions over P_S and C, and the action set $\mathcal{A}$ is the set of possible ground atoms over P_A and C (for more details, please refer to [14,7]).

When $\mathcal{S}$ and $\mathcal{A}$ in a relational MDP are ground sets, we call this a *concrete* MDP. Note that a relational representation enables us to aggregate states and actions by using variables instead of constants in the predicate terms. A *substitution* θ is a set $\{\mathtt{X}_1/\mathtt{t}_1, \ldots, \mathtt{X}_n/\mathtt{t}_n\}$, binding each variable $\mathtt{X}_i$ to a term $\mathtt{t}_i$.

We call *abstract state* σ (and *abstract action* α) the representation in which all the constants of a ground state s (and a ground action a) are replaced by variables. In this work we consider only the level of abstraction that preserves every predicade in the conjunction without any constants in the terms. We denote by $\mathcal{S}_\sigma$ the set of ground states covered by an abstract state σ. Similarly we define $\mathcal{A}_\alpha$ as the set of all ground actions covered by an abstract action α. We also define $\mathcal{S}_{ab}$ and $\mathcal{A}_{ab}$ as the set of all abstract states and the set of all abstract actions in a relational representation of an MDP, respectively. Consider an abstract state $\sigma = \{p_1(X_1), p_2(X_1, X_2)\}$, $\sigma \in \mathcal{S}_{ab}$; a ground state $s_1 \in \mathcal{S}$, $s_1 = \{p_1(t_1), p_2(t_1, t_2)\}$ is abstracted by σ with $\theta = \{\mathtt{X}_1/\mathtt{t}_1, \mathtt{X}_2/\mathtt{t}_2\}$ and a ground state $s_2 \in \mathcal{S}$, $s_2 = \{p_1(t_3), p_2(t_3, t_4)\}$ is abstracted by σ with $\theta = \{\mathtt{X}_1/\mathtt{t}_3, \mathtt{X}_2/\mathtt{t}_4\}$. In this case σ represents an abstraction of both s_1 and s_2, and $s_1, s_2 \in \mathcal{S}_\sigma$.

An abstract policy π_{ab} specifies a mapping from abstract states into abstract actions: $\pi_{ab} : \mathcal{S}_{ab} \times \mathcal{A}_{ab} \rightarrow [0, 1]$. The challenge is to apply an abstract policy in a concrete problem: we must provide a way to translate from the concrete to the abstract level, and vice-versa. Problems that can use the same predicates to describe its states and actions are in the same *domain class*; that is, problems that have the same spaces of abstract states $\mathcal{S}_{ab}$ and abstract actions $\mathcal{A}_{ab}$ are in the same domain class. When the set of objects and a transition function are added, we specifie a *domain* $\mathcal{D}$. It describes the dynamics of the world and also the number of states and actions are now fixed. Finally, a *task* Ω is a tuple $\langle \mathcal{D}, r, \mathcal{G}, \beta \rangle$, where $\mathcal{D}$ is a domain, r is the reward function, $\mathcal{G}$ is the set of goal states, that indicates which states of the domain are desirable and β is the probability distribution over initial states. A task fully describes an MDP. Furthermore, in this work we focus on the transfer learning among different tasks within the same domain class.

Now we can explain how to use an abstract policy in a concrete MDP. Assume a stochastic abstract policy π_{ab} is given. For a ground state s we find the corresponding abstract state σ so that $s \in \mathcal{S}_\sigma$, *i.e.*, $\xi_s(s) = \sigma$. The current stochastic abstract policy π_{ab} defines a probability distribution over abstract actions $\mathcal{A}_{ab}$ to be applied in the abstract state $\sigma \in \mathcal{S}_{ab}$. We then choose probabilistically the abstract action $\alpha \in \mathcal{A}_{ab}$ to be executed in σ. Note that an abstract action α defines $\mathcal{A}_\alpha$; thus, applying the abstract policy π_{ab} results in the mapping from an abstract state into a set of ground actions. To produce a particular ground action, we define $\xi_a : \mathcal{A}_{ab} \rightarrow \mathcal{A}$. Here we define ξ_a as a function that randomly selects (with uniform probability) a concrete action a from the set of concrete actions $\mathcal{A}_\alpha \cap \mathcal{A}^s$, which combines $\mathcal{A}_\alpha$ with the set $\mathcal{A}^s$ of allowable actions in state $s \in \mathcal{S}$. Obviously, other schemes may be used to produce a concrete action a. This whole process yields the grounding of a stochastic abstract policy $\pi_{ab}(\xi_s(s), \alpha)$, denoted by $a = \text{grounding}_{\pi_{ab}}(\alpha, s)$.

3.2 Solving MDPs with AbsProb-PI

The task of the agent in an MDP is to find a policy. In a concrete fully-observable MDP, the set of deterministic memoryless policies, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, contains an optimal policy [16,10]. When considering abstract states and actions, stochastic

policies are appropriate, because as they are more flexible by offering more than one choice of action per state, they can be arbitrarily better than deterministic policies [18,17]. We define a *memoryless stochastic abstract policy* as $\pi_{ab} : \mathcal{S}_{ab} \times \mathcal{A}_{ab} \rightarrow [0, 1]$, with $P(\alpha|\sigma) = \pi_{ab}(\sigma, \alpha)$, $\sigma \in \mathcal{S}_{ab}$, $\alpha \in \mathcal{A}_{ab}$.

AbsProb-PI [17] is an algorithm that, given an MDP, uses cumulative discounted reward evaluation to build an abstract policy. It is a model-based algorithm, i.e., it requires prior knowledge of the whole model, including transition and reward functions. AbsProb-PI is a Policy Iteration algorithm, designed to perform in an abstract level. At each iteration, a gradient function G is used to determine the improvement direction of the current policy. As the abstract state space $\mathcal{S}_{ab}$ does not necessarily hold the Markov property, AbsProb-PI considers the initial state distribution $\beta(s_0)$. Two parameters must be defined: $\epsilon > 0$ which guarantees that the policy converges at most to an ϵ-greedy policy; and $\rho(i)$, the step size used in the gradient descent method for each iteration i. We refer to [17] for a thorough presentation of AbsProb-PI.

The use of a relational representation enables us to generalize experiences and to define abstract policies, making it easier to transfer knowledge between tasks in different domains. Abstract policies can effectively be used in new similar problems, or even be used to accelerate reinforcement learning [14,7], as we show in Section 4.

3.3 Transferring Risk-Aware Abstract Policies

To achieve transfer learning, we first need to decide which policy will be transferred. Policies found using different risk attitudes result in different performances (i.e., history size) of the agent. Figure 1 shows the probability as a function of history size $|h|$ (number of steps to reach goal) given discount factors $\gamma = 1.0$ and $\gamma = 0.9$ for several tasks (i.e., several goal states) on the navigation environment shown in Figure 2-left (see section 4 for the environment description). This figure shows that: (i) if γ is small (risk-prone attitude) there is a high probability for short histories, but also high probabilities for very long histories (histories with $|h| > 200$ steps accumulate almost half of the histories); and (ii) if γ is large (risk-neutral attitude) there is a medium probability for short histories, but also low probability for very long histories (histories with $|h| > 200$ steps happen with probability lower than 0.02).

We have options: we can transfer a risk-prone policy π_{RP} (found with discount factor γ_{RP}), or a risk-neutral policy π_{RN} (found with γ_{RN}, with $\gamma_{RP} < \gamma_{RN}$). Under π_{RP} there are higher probabilities for short histories, whereas under π_{RN} there are lower probabilities for longer histories. Our strategy is to combine both. It starts by applying π_{RP}, and then changes to π_{RN} as time goes on. The transition between policies is done linearly with time, i.e., let π^t_{NS} be the abstract policy applied at time step t, then the non-stationary policy π^t_{NS} is defined by:

$$\pi^t_{NS}(\sigma, \alpha) = (1 - \mu(t))\pi_{RP}(\sigma, \alpha) + \mu(t)\pi_{RN}(\sigma, \alpha) \; \forall \sigma \in \mathcal{S}_{ab}, \forall \alpha \in \mathcal{A}_{ab},$$

with $\mu(t) = \min\left\{\frac{t}{thr}, 1\right\}$.

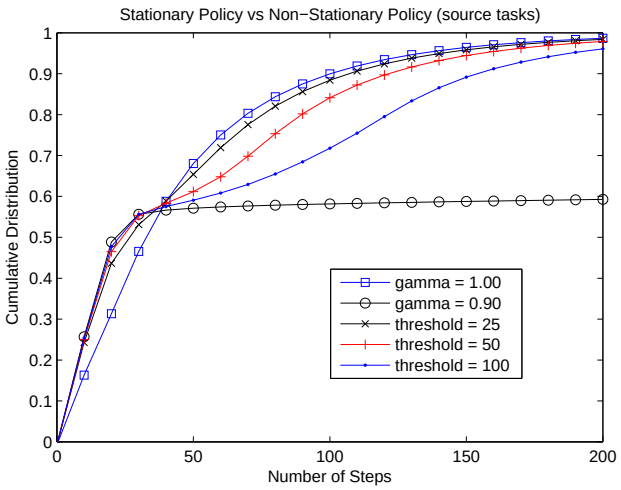

Fig. 1: Probability of success versus history size of stationary policies found with different γ values and non-stationary policies with different threshold values.

The parameter thr is such that π_{RN} is fully applied for $t > thr$. Figure 1 also shows the performance of such strategy under different values of thr; our strategy is applied on source tasks, in which the policies π_{RP} and π_{RN} were found. If threshold parameter thr is well tuned, the non-stationary policy π_{NS} shows a better balance between short histories and long histories.

Now that we have a policy to transfer, we have to reuse this policy when learning a new task. This policy is used preferably to guide the exploration of the space state. Obviously, other approaches can be used.

4 Experiments

We conducted experiments to evaluate our proposal; that is, to find and use a risk-aware abstract policy in inter-task transfer applications. All experiments were made on a simulated robotic navigation domain, described below.

Navigation Domain Class

We use a robotic navigation domain class, in which the task of the agent is to navigate through an environment to reach a specific location. The space is divided in several cells of equal size, each representing a single state. Besides, the results of actions are probabilistic: if the agent performs a movement action, there is a probability p (we use $p = 0.9$) that it succeeds (and thus changes state), while it does not move with probability $1 - p$. Figure 2 shows instances of this domain.

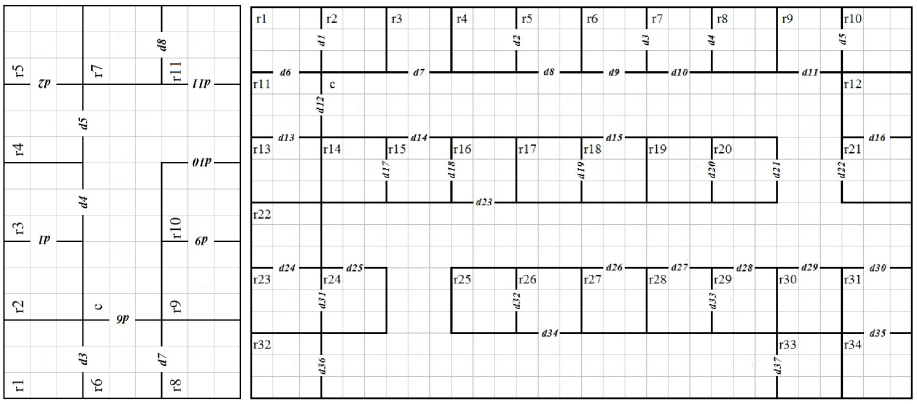

Fig. 2: The robotic navigation domain class. Left: an environment with 11 rooms (*source domain*). Right: an environment with 34 rooms (*target domain*).

There are four types of objects the agent can recognize: rooms, doors, markers and corridors. Additionally, it has a positioning system as well, being able to tell if it is near or far from the goal location. Therefore, we describe each ground state and action using the relational alphabet $C \cup P_S \cup P_A$. C is the set of objects. P_S describes predicates related to observations, $P_S = \{\texttt{inRoom}(R), \texttt{inCorridor}(C), \texttt{seeDoorFar}(D), \texttt{seeAdjRoom}(D), \texttt{seeAdjCorridor}(D), \texttt{seeEmptySpace}(M), \texttt{nearGoal}, \texttt{farGoal}, \texttt{appGoal}(X), \texttt{awayGoal}(X)\}$, where R is a room, C is a corridor, M is a marker, D is a door and X is any object. The range of vision of the robot is of one cell for markers, but it can perceive doors that are two cells distant. The predicate `seeDoorFar` means that the robot can see a door, but it is at least two cells distant, whereas $\texttt{seeAdjRoom}(D)$ and $\texttt{seeAdjCorridor}(D)$ indicate that the robot is so close to that door (one cell distant) that it can even see what lies through it (if it is a room or a corridor). $\texttt{seeEmptySpace}(M)$ means that the robot sees a free space with a marker, where it could move to. `nearGoal` is true if the robot is at a close distance to the goal, i.e., at a Manhattan distance to the goal smaller than 5 and `farGoal` is true if the robot is at a far distance from the goal, i.e., at a Manhattan distance to the goal bigger than 8. There are indications of two possible directions: $\texttt{appGoal}(X)$ (approaching goal) or $\texttt{awayGoal}(X)$ (moving away from the goal) and these predicates indicates that the object X is closer or farther than the goal in relation to the agent.

All actions the agent can take moves the agent toward an object, with a step of one cell (unless it stays in the same state due to environment dynamics). Agent also has to consider whether it is approaching the goal or moving away from it. That being said, the predicates for actions are:

$$
\begin{aligned}
P_A = \{\ & \texttt{goToDoorAppGoal}(di), \texttt{goToDoorAwayGoal}(di), \\
& \texttt{goToRoomAppGoal}(ri), \texttt{goToRoomAwayGoal}(ri), \\
& \texttt{goToCorridorAppGoal}(ci), \texttt{goToCorridorAwayGoal}(ci), \\
& \texttt{goToEmptyAppGoal}(mi), \texttt{goToEmptyAwayGoal}(mi)\}.
\end{aligned}
$$

Each task has a unique goal state ($\mathcal{G} = \{s_G\}$), which in our experiments is always the cell in the center of a room. The experiments conditions and their results are discussed in the next section.

Results

We use two domains in our experiments, both of which are represented in Figure 2. The first one, the source domain, contains 11 rooms and the center of all of them is used as the goal state for a task, resulting in a total of 11 tasks. The second one, the target domain, contains 34 rooms, and similarly, 34 tasks. Considering all 11 tasks together in the source domain, the agent groups them as a single big task and then uses AbsProb-PI with $\gamma = 0.9$ and $\gamma = 1.0$ to find, respectively, abstract policies π_{RP} and π_{RN}. Parameters used in AbsProb-PI were: $\rho(i) = \frac{1}{1+0.5i}$, $\epsilon = 0.05$, initial state distribution is uniform for all states, and stop criterion after 500 iterations. With these two learned policies at hand, a non-stationary policy π_{NS} is also created after setting $thr = 50$. To assess if the non-stationary policy performs better than policies π_{RP} and π_{RN} when applying in different tasks (transfer), we execute them in each task of the target domain, one at a time. Figure 3-left shows the probability of success versus histories size $|h|$ when applying these three policies. We also compare against random policy π_{rand} as a reference point.

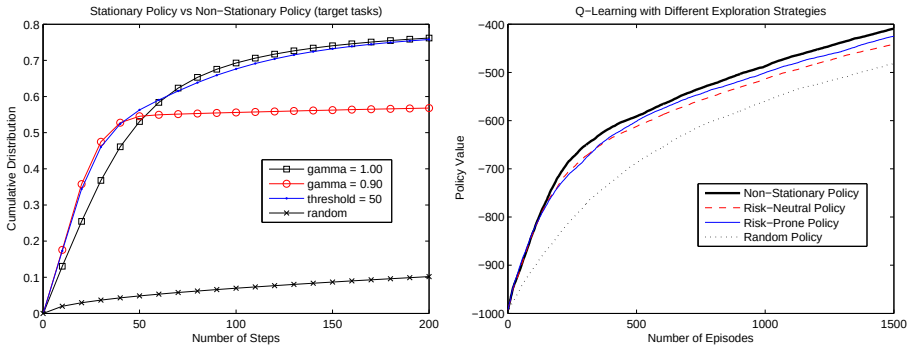

Fig. 3: Left: Cumulative probability distribution over history size of π_{RP}, π_{RN} and π_{NS}. Right: Transfer learning with the combined policy π_{NS} compared to learning reusing π_{RP}, π_{RN} and π_{random}.

One can notice that the non-stationary policy indeed explores the best of the two stationary policies and all of them are substantially better than the random policy.

We now evaluate the effectiveness of transfer learning when using risk-prone, risk-neutral, non-stationary, and random policies. On each of the 34 tasks in the target domain, an agent learns a policy using the Q-Learning algorithm with an

ϵ-greedy exploitation/exploration strategy. We always use the transferred policy in the exploration phase. We used the random policy in the exploration phase as traditionally is done in the ϵ-greedy strategy and compared the results when replacing it by a past abstract policy (π_{RP}, π_{RN} or π_{NS}). It is worth mentioning that the abstract policies are also stochastic and all actions have a minimum probability of ϵ, hence exploration is guaranteed. We set learning rate to 0.05 for the Q-Learning algorithm, $\epsilon = 0.1$ and $\gamma = 0.999$. Each episode has maximum number of steps 1,000 and learning was done for 15,000 episodes; each task in the target domain was learned 5 times.

Results for transfer learning are shown on Figure 3-right that plots the current policy value versus the number of steps. We observe that the non-stationary policy outperforms all of them.

5 Conclusions

In this paper we have proposed algorithms for a risk-aware robot that must transfer abstract policies from already solved tasks to new tasks. Our abstract policies employ relational representations, and our policies are memoryless, hence compact, and stochastic. We have derived the necessary algorithms, and our experiments demonstrate that our methods are effective in navigation problems.

We have presented a formulation of risk-aware SSP problems (with constant negative reward and unique goal state) that reduces risk-awareness to γ-tuning; our analysis is novel in that it justifies even values of γ that are larger than one. We then used these insights to combine abstract policies learned by the AbsProb-PI algorithm. This algorithm generates policies under the infinite-horizon discounted criterion; by appropriately changing γ, we obtain risk-aware behavior out of AbsProb-PI. We have proposed a method that combines linearly the various abstract policies generated by AbsProb-PI, emphasizing a risk-prone behavior in the first stages of learning, and a risk-neutral behavior in latter stages. The resulting combined policy is not only stochastic, but also non-stationary. Our experiments have focused on a robot in indoor navigation, and have shown that a robot that mixes the combined policy with learning in a new task does have an advantage over a robot learning from scratch a new policy in the new task.

Our contributions are both an extended interpretations for γ-tuning as risk-awareness, and a method that employs risk-awareness in transfer learning. It is necessary still to better understand the risk-averse behavior that obtains when $\gamma > 1$ is employed; we plan to focus on this development in the future.

References

1. Banerjee, B., Stone, P.: General game learning using knowledge transfer. In: Proc. of the twentieth Int. Jt. Conf. on Artif. Intell. pp. 672–677. AAAI Press (2007)
2. Bertsekas, D.P., Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. Math. of Oper. Res. 16(3), 580–595 (1991)

3. Bianchi, R., Ribeiro, C., Costa, A.: Accelerating autonomous learning by using heuristic selection of actions. J. of Heuristics 14, 135–168 (2008)
4. Delage, E., Mannor, S.: Percentile optimization for markov decision processes with parameter uncertainty. Oper. Res. 58(1), 203–213 (2010)
5. Fernández, F., García, J., Veloso, M.: Probabilistic Policy Reuse for inter-task transfer learning. Robotics and Auton. Syst. 58(7), 866–871 (Jul 2010)
6. Howard, R.A., Matheson, J.E.: Risk-sensitive markov decision processes. Management Science 18(7), 356–369 (1972)
7. Koga, M.L., Silva, V.F., Costa, A.H.R.: Speeding-up reinforcement learning tasks through abstraction and transfer learning. In: Proc. of the twelfth Int. Jt. Conf. on Auton. Agents and Multiagent Syst. (AAMAS '13). pp. 119–126 (2013)
8. Konidaris, G., Scheidwasser, I., Barto, A.: Transfer in reinforcement learning via shared features. J. of Mach. Learn. Res. 13, 1333–1371 (2012)
9. Li, L., Walsh, T.J., Littman, M.L.: Towards a Unified Theory of State Abstraction for MDPs. In: Proc. of the ninth Int. Sympos. on Artif. Intell. and Math. pp. 531–539. ISAIM (2006)
10. Littman, M.L.: Memoryless policies: theoretical limitations and practical results. In: Proc. of the third Int. Conf. on Simul. of Adapt. Behav.: from animals to animats 3. pp. 238–245. MIT Press, Brighton (1994)
11. Liu, Y., Koenig, S.: Probabilistic planning with nonlinear utility functions. In: ICAPS. pp. 410–413 (2006)
12. Liu, Y., Stone, P.: Value-function-based transfer for reinforcement learning using structure mapping. In: Proc. of the twenty-first Natl. Conf. on Artif. Intell. pp. 415–420. AAAI Press (2006)
13. Mannor, S., Tsitsiklis, J.: Mean-variance optimization in markov decision processes. In: Proc. of the twenty-eighth Intl. Conf. on Mach. Learn. (ICML '11). pp. 177–184. ACM (2011)
14. Matos, T., Bergamo, Y.P., Silva, V.F., Cozman, F.G., Costa, A.H.R.: Simultaneous Abstract and Concrete Reinforcement Learning. In: Proc. of the ninth Symp. of Abstr., Reformul., and Approx. (SARA'11). pp. 82–89. AAAI Press (2011)
15. Minami, R., da Silva, V.F.: Shortest stochastic path with risk sensitive evaluation. In: Proc. of the eleventh Mexican Int. Conf. on Artif. Intell. (MICAI'12). pp. 370–381. Springer (2012)
16. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc. (1994)
17. Silva, V.F., Pereira, F.A., Costa, A.H.R.: Finding memoryless probabilistic relational policies for inter-task reuse. In: Proc. of the fourteenth Int. Conf. on Inf. Process. and Manag. of Uncertain. (IPMU'12). Communications in Computer and Information Science, vol. 298, pp. 107–116. Springer (2012)
18. Singh, S.P., Jaakkola, T., Jordan, M.I.: Learning without state-estimation in partially observable markovian decision processes. In: Proc. of the eleventh Int. Conf. on Mach. Learn. (ICML '94). vol. 31, p. 37. Morgan Kaufmann (1994)
19. Taylor, M.E., Stone, P., Liu, Y.: Transfer learning via inter-task mappings for temporal difference learning. J. of Mach. Learn. Res. 8(1), 2125–2167 (2007)
20. Wasserman, L.: All of Statistics: A Concise Course in Statistical Inference. Springer (2003)
21. Whittle, P.: Why discount? the rationale of discounting in optimisation problems. In: Heyde, C., Prohorov, Y., Pyke, R., Rachev, S. (eds.) Athens Conference on Applied Probability and Time Series Analysis, Lecture Notes in Statistics, vol. 114, pp. 354–360. Springer New York (1996)