Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling

Short-Sighted Stochastic Shortest Path Problems

Felipe W. Trevizan
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA, USA
fwt@cs.cmu.edu

Manuela M. Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
mmv@cs.cmu.edu

Abstract

Algorithms to solve probabilistic planning problems can be classified in probabilistic planners and replanners. Probabilistic planners invest significant computational effort to generate a closed policy, i.e., a mapping function from every state to an action, and these solutions never "fail" if the problem correctly models the environment. Alternatively, replanners computes a partial policy, i.e., a mapping function from a set of the state space to an action, and when and if such policy fails during execution in the environment, the replanner is re-invoked to plan again from the failed state. In this paper, we introduce a special case of Stochastic Shortest Path Problems (SSPs), the short-sighted SSPs, in which every state has positive probability of being reached using at most t actions. We introduce the novel algorithm Short-Sighted Probabilistic Planner (SSiPP) that solves SSPs through short-sighted SSPs and guarantees that at least t actions can be executed without replanning. Therefore, by varying t, SSiPP can behave as either a probabilistic planner by computing closed policies, or a replanner by computing partial policies. Moreover, we prove that SSiPP is asymptotically optimal, making SSiPP the only planner that, at the same time, guarantees optimality and offers a bound in the minimum number of actions executed without replanning. We empirically compare SSiPP with the winners of the previous probabilistic planning competitions and, in 81.7% of the problems, SSiPP performs at least as good as the best competitor.

1 Introduction

Probabilistic planning captures the uncertainty of plan execution by probabilistically modeling the effects of actions in the environment, and therefore the probability of reaching different states from a given state and action. As commonly used, we consider a probabilistic planning problem to be represented as a Stochastic Shortest Path Problem (SSP) (Bertsekas and Tsitsiklis 1991), where an initial state and a set of goals states are given and actions have predefined probabilistic transition among states. In this work, we address the question of how to *efficiently and optimally solve* probabilistic planning problems.

One approach to solve probabilistic planning problems is to use value iteration and policy iteration algorithms, which are optimal algorithms (Bertsekas and Tsitsiklis 1996). Planners based on these algorithms return a closed policy, i.e., an universal mapping function from every state to the optimal action that leads to a goal state. Assuming the model correctly captures the cost and uncertainty of the actions in the environment, these solutions to a probabilistic planning problem are extremely powerful as the execution of such policies never "fails," and the planner does not need to be re-invoked ever. Unfortunately the computation of such policies is prohibitive in complexity as problems scale up. The efficiency of value-iteration based probabilistic planners can be improved by combining asynchronous updates and heuristic search (e.g., Labeled RTDP (Bonet and Geffner 2003)), resulting in optimal algorithms with convergence bounds. Although these techniques allow planners compute compact policies, in the worst case these policies are still linear in the size of the state space, which itself can be exponential in the size of the state or goals.

Another approach to solve probabilistic planning problems is by replanning. Replanners, do not invest the computational effort to generate a closed policy, and instead compute a partial policy, i.e. a policy that does not address all the probabilistic possible reachable states. Different methods can be employed to generate partial policies, e.g. determinization (Yoon, Fern, and Givan 2007; Yoon et al. 2008), sampling (Dean et al. 1995; Teichteil-Koenigsbuch, Infantes, and Kuter 2008) and finite horizon search (Pearl 1985; Kocsis and Szepesvri 2006). During the execution of such solution in the environment, a state not included in the partial policy can be reached and in this case the replanner is re-invoked to compute a new partial policy starting from the unpredicted state.

Interestingly, replanning approaches have shown to perform well at the International Probabilistic Planning Competition (IPPC) (Younes et al. 2005; Bonet and Givan 2007; Bryce and Buffet 2008), e.g., with the winning performance of FF-Replan (Yoon, Fern, and Givan 2007). The idea behind FF-Replan is simple and powerful: relax the probabilistic problem into a deterministic problem D and use the deterministic planner FF (Hoffmann and Nebel 2001) to solve D. When and if the execution of the solution for D fails in the probabilistic environment, FF is re-invoked to plan again from the failed state. Probabilistic planners at IPPC did not perform as well since they could not computation-

ally solve large problems. Despite its major success, FF-Replan is oblivious to probabilities and dead-ends, leading to poor performance in specific IPPC problems, e.g., the triangle tire-domain (Little and Thiébaux 2007).

In this work, we present an approach that offers the best from both the probabilistic planners and the replanners, namely optimal solution generation, and the efficient incomplete solution generation, respectively.

Specifically, we define a new probabilistic model, the short-sighted Stochastic Shortest Path Problems (short-sighted SSPs), a special case of SSPs in which every state has positive probability of being reached using at most t actions. We introduce the novel algorithm Short-Sighted Probabilistic Planner (SSiPP) that solves SSPs through short-sighted SSPs and guarantees that at least t actions can be executed without replanning. Therefore, by varying t, SSiPP can behave as either a probabilistic planner by computing closed policies, or a replanner by computing partial policies. Moreover, we prove that SSiPP is asymptotically optimal, making SSiPP the only planner that, at the same time, guarantees optimality and offers a bound in the minimum number of actions executed without replanning.

The remainder of this paper is organized as follows: Section 2 reviews related work and Section 3 introduces the basic concepts and notation. Section 4 defines formally short-sighted SSPs. Section 5 presents our main theoretical results for short-sighted SSPs. Section 6 presents the SSiPP algorithm and the proof of its asymptotically optimal performance. Section 7 empirically evaluates the impact of t in the time to compute ϵ-approximations to the optimal solution and compares SSiPP with the winners of the previous IPPCs. Section 8 concludes the paper.

2 Related Work

FF-Hindsight (Yoon et al. 2008) is a non-optimal replanner that generalizes FF-Replan and consists of three steps: (i) randomly generate a set of non-stationary deterministic problems D starting from s; (ii) use FF to solve them; and (iii) combine the cost of their solution to estimate the true cost of reaching a goal state from s. Each deterministic problem in D has a fixed length (horizon) and is generated by sampling one outcome of each probabilistic action for each time step. This process reveals two major drawbacks of FF-Hindsight: (i) a bound in the horizon size of the problem is needed in order to produce the relaxed problems; and (ii) rare effects of actions might be ignored by the sampling procedure. While the first drawback is intrinsic to the algorithm, a workaround to the second one is proposed (Yoon et al. 2010) by always adding the all-outcomes relaxation of the problem to D and, therefore, ensuring that every effect of an action appears at least in one deterministic problem in D.

Also based on sampling, the Upper Confidence bound for Trees (UCT) algorithm (Kocsis and Szepesvri 2006) is a non-optimal replanner that chooses actions greedily according to an approximation of the t-look-ahead heuristic (Pearl 1985). Given $t > 0$, this approximation is obtained by solving a series of t *multi-armed bandits* problems where each *arm* represents an action and a finite-horizon of t actions are considered. Sparse sampling techniques are employed to efficiently solve this new problem.

Another relevant approach is performed by two non-optimal replanners: Envelope Propagation (EP) (Dean et al. 1995) and Robust FF (RFF) (Teichteil-Koenigsbuch, Infantes, and Kuter 2008). In general terms, EP and RFF compute an initial partial policy π and iteratively expand it in order to avoid replanning. EP prunes the state space S and represents the removed states by a special meta state `out` and the appropriate meta actions to represent the transitions from and to `out`. At each iteration, EP refines its approximation S' of S by expanding and then re-pruning S'. Re-pruning is necessary to avoid the convergence of S' to S and this is the main drawback of EP because low probability states are pruned and therefore ignored. As in FF-Replan, these states might be necessary in order to avoid large costs and dead-ends. RFF uses a different approach: an initial partial policy π is computed by solving the most-likely outcome determinization of the original problem using FF and then the *robustness* of π is iteratively improved. For RFF, robustness is a synonym of probability of replanning, i.e., given $\rho \in [0, 1]$, RFF computes π such that the probability of replanning when executing π from s_0 is at most ρ. An approach similar to RFF and EP in the context of motion planning is Variable Level-of-detail Motion Planner (Zickler and Veloso 2010) in which poorly predictable physical interactions are ignored (pruned) in the far future.

The first asymptotically optimal replanner proposed by the community, Real Time Dynamic Programming (RTDP) (Barto, Bradtke, and Singh 1995), is a heuristic search algorithm that interleaves search and execution. Given an admissible heuristic, RTDP maintains a lower bound for V^* and acts greedily according to it. In order to avoid being trapped in loops and to find the optimal solution, RTDP updates the lower bound of every state visited during its search. If a goal state is reachable from every other state and the input heuristic is admissible, then after several executions of RTDP (possibly infinitely many), its lower bound converges to V^* over the relevant states. Unlike the replanners described so far that have no optimality guarantee, RTDP is asymptotically optimal, however no guarantee is given w.r.t. to replanning. The scalability of RTDP is improved by ReTrASE (Kolobov, Mausam, and Weld 2009) in which the lower bound on V^* is projected into a lower dimensional space. The set of base functions used by ReTrASE is obtained by solving the all outcomes determinization of the original problem using FF. Due to the lower dimensional representation, ReTrASE is a non-optimal replanner.

Although RTDP is a replanner, it is mainly used as a probabilistic planner: RTDP simulates the problem internally until the lower bound on V^* has approximately converged and a closed policy is extracted from it. Based on this idea, several extensions of RTDP were proposed and, due to space constraints, we only highlight some of those. Labeled RTDP (LRTDP) (Bonet and Geffner 2003) introduces a labelling schema to find states that have already converged and therefore avoid exploring them again. With this technique, LRTDP provides an upper bound on the number of iterations necessary to reach ϵ-convergence w.r.t. V^*. Two

other algorithms, Bounded RTDP (McMahan, Likhachev, and Gordon 2005) and Focused RTDP (Smith and Simmons 2006) provide empirical evidence of improvements on the convergence time of RTDP by also keeping track of an upper bound on V^*.

SSiPP, the planner presented in this paper, differs from all the planners presented above by, at the same time, being asymptotically optimal and offering a bound in the minimum number of actions executed without replanning.

3 Background

A Stochastic Shortest Path Problem (SSP) is defined by the tuple $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^{\mathsf{G}} \rangle$, in which (Bertsekas and Tsitsiklis 1991; Bonet and Geffner 2003):[1]

- S is the finite set of state;
- $s_0 \in \mathsf{S}$ is the initial state;
- $\mathsf{G} \subseteq \mathsf{S}$ is the set of goal states;
- A is the finite set of actions;
- $P(s'|s,a)$ represents the probability that $s' \in \mathsf{S}$ is reached after applying action $a \in \mathsf{A}$ in state $s \in \mathsf{S}$;
- $C(s,a) \in (0, +\infty)$ is the cost of applying action $a \in \mathsf{A}$ in state s; and
- $C^{\mathsf{G}} \colon \mathsf{G} \to [0, +\infty)$ represents the cost incurred when the given goal state is reached.

A solution to an SSP is a policy π, i.e., a mapping from S to A. If π is defined over the entire space S, then π is a closed policy. A policy π defined only for the states reachable from s_0 when following π is a closed policy w.r.t. s_0 and $\mathsf{S}(\pi, s_0)$ denotes this set of reachable states. For instance, in the SSP depicted in Figure 1, the policy $\pi_0 = \{(s_0, a_0), (s'_1, a_0)\}$ is a closed policy w.r.t. s_0 and $\mathsf{S}(\pi_0, s_0) = \{s_0, s'_1, s_{\mathsf{G}}\}$.

Given a policy π, we define trajectory as a sequence $\mathcal{T}_\pi = \langle s_{(0)}, \ldots, s_{(k)} \rangle$ such that, for all $i \in \{0, \cdots, k-1\}$, $\pi(s_{(i)})$ is defined and $P(s_{(i+1)}|s_{(i)}, \pi(s_{(i)})) > 0$. An optimal policy π^* is any policy that always reaches a goal state when followed from s_0 and also minimizes the expected cost of $\mathcal{T}_{\pi^*}$. For a given SSP, π^* might not be unique, however the optimal value function V^*, i.e., the mapping from states to the minimum expected cost to reach a goal state, is unique. V^* is the fixed point of the set of equations defined by (1) for all $s \in \mathsf{S} \setminus \mathsf{G}$ and $V^*(s) = C^{\mathsf{G}}(s)$ for all $s \in \mathsf{G}$.

$$V^*(s) = \min_{a \in \mathsf{A}} C(s,a) + \sum_{s' \in \mathsf{S}} P(s'|s,a) V^*(s') \qquad (1)$$

Definition 1 (reachability assumption). *An SSP satisfies the reachability assumption if, for all $s \in \mathsf{S}$, there exists a policy π and a trajectory $\mathcal{T}_\pi = \langle s, \ldots, s_{(k)} \rangle$, s.t. $s_{(k)} \in \mathsf{G}$.*

Given an SSP $\mathbb{S}$, if a goal state can be reached with positive probability from every state $s \in \mathsf{S}$, then the reachability assumption (Definition 1) holds for $\mathbb{S}$ and $0 \leq V^*(s) < \infty$

[1] SSPs are closely related to Markov Decision Processes (MDPs) (Puterman 1994) and SSPs are strictly more expressive than infinite-horizon MDPs (Bertsekas 1995).

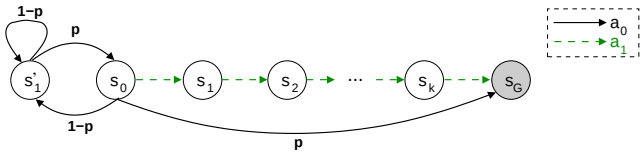

Figure 1: Example of an SSP. The initial state is s_0, the goal state is s_{G}, $C(s,a) = 1, \forall s \in \mathsf{S}, \forall a \in \mathsf{A}$ and $C^{\mathsf{G}}(s_{\mathsf{G}}) = 0$.

(Bertsekas 1995). Once V^* is known, any optimal policy π^* can be extracted from V^* by substituting the operator $\min$ by argmin in equation (1).

A possible approach to compute V^* is the value iteration algorithm: define $V^{i+1}(s)$ as in (1) with V^i in the right hand side instead of V^* and the sequence $\langle V^0, V^1, \ldots, V^k \rangle$ converges to V^* as $k \to \infty$ (Bertsekas 1995). The process of computing V^{i+1} from V^i is known as Bellman update and $V^0(s)$ can be initialized with an admissible heuristic $H(s)$, i.e., a lower bound for V^*. In practice we are interested in reaching ϵ-convergence, that is, given ϵ, find V such that $\max_s |V(s) - \min_a C(s,a) + \sum_{s'} P(s'|s,a)V(s')| \leq \epsilon$. We conclude this section with the following well-known result necessary in most of our proofs (Bertsekas and Tsitsiklis 1996, Assumption 2.2 and Lemma 2.1):

Theorem 1. *Given an SSP $\mathbb{S}$, if the reachability assumption holds for $\mathbb{S}$, then the admissibility and monotonicity of V are preserved through the Bellman updates.*

4 Short-Sighted Stochastic Shortest Path Problems

Several replanners address the scalability issue by simplifying the action space through the all outcomes or most likely outcome determinization. The drawback of this approach is being oblivious to the probability of each outcome of actions and their correlation. We introduce *short-sighted* SSPs, a model that simplifies the state space instead of the action space of a given SSP.

Definition 2 ($\delta(s,s')$). *The non-symmetric distance $\delta(s,s')$ between two states s and s' is $\operatorname{argmin}_k \{\mathcal{T}_\pi = \langle s, s_{(1)}, \ldots, s_{(k-1)}, s' \rangle | \exists \pi$ and $\mathcal{T}_\pi$ is a trajectory$\}$.*

Definition 3 (Short-Sighted SSP). *Given an SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^{\mathsf{G}} \rangle$, a state $s \in \mathsf{S}$, $t > 0$ and a heuristic H, the (s,t)-short-sighted SSP $\mathbb{S}_{s,t} = \langle \mathsf{S}_{s,t}, s, \mathsf{G}_{s,t}, \mathsf{A}, P, C, C^{\mathsf{G}}_{s,t} \rangle$ associated with $\mathbb{S}$ is defined as:*

- $\mathsf{S}_{s,t} = \{s' \in \mathsf{S} | \delta(s,s') \leq t\}$;
- $\mathsf{G}_{s,t} = \{s' \in \mathsf{S} | \delta(s,s') = t\} \cup (\mathsf{G} \cap \mathsf{S}_{s,t})$;
- $C^{\mathsf{G}}_{s,t}(s) = \begin{cases} C^{\mathsf{G}}(s) & \text{if } s \in \mathsf{G} \\ H(s) & \text{if } s \in \mathsf{G}_{s,t} \setminus \mathsf{G} \end{cases}$

For simplicity, when H is not clear by context nor explicit, then $H(s) = 0$ for all $s \in \mathsf{S}$.

Figure 2 shows the $(s_0, 2)$-short-sighted SSP associated with the example in Figure 1. The state space $\mathsf{S}_{s,t}$ of (s,t)-short-sighted SSPs is a subset of the original state space in

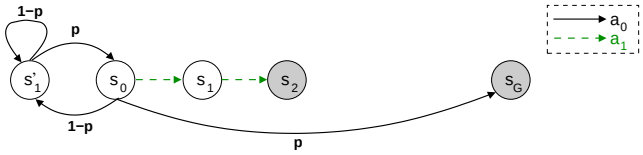

Figure 2: The $(s_0, 2)$-short sighted SSP associated to the SSP in Figure 1. The set of goal states $\mathsf{G}_{s_0,2}$ is $\{s_\mathsf{G}, s_2\}$, $C^\mathsf{G}_{s_0,2}(s_\mathsf{G}) = 0$ and $C^\mathsf{G}_{s_0,2}(s_2) = H(s_2) = 0$.

which any state $s' \in \mathsf{S}_{s,t}$ is reachable from s using at most t actions. The key property of short-sighted SSPs that allows them to be used for solving SSPs is given by the definition of $C^\mathsf{G}_{s,t}$: every artificial goal state s'_G, i.e. $s'_\mathsf{G} \in \mathsf{G}_{s,t} \setminus \mathsf{G}$, has its heuristic value $H(s'_\mathsf{G})$ used as goal cost $C^\mathsf{G}_{s,t}(s'_\mathsf{G})$. Therefore, the search for a solution to short-sighted SSPs is guided towards the goal states of the original SSP, even if such states are not in $\mathsf{G}_{s,t}$.

The optimal value function for $\mathbb{S}_{s,t}$ on s, $V^*_{\mathbb{S}_{s,t}}(s)$, is related to the value obtained by the t-look-ahead heuristic $L_t(s)$ for s (Pearl 1985). As an intuition of the difference between using L_t and $V^*_{\mathbb{S}_{s,t}}$ as heuristics to solve SSPs, consider the example depicted in Figure 1 and depth $t = 2$:

- $L_2(s_0)$ is the minimum expected cost of executing 2 actions in a row, therefore only trajectories of size 2 are considered; and
- $V^*_{\mathbb{S}_{s_0,2}}(s_0)$ is the minimum expected cost to reach a goal state in $\mathbb{S}_{s_0,2}$ (Figure 2) from s_0. Thus all possible trajectories in $\mathbb{S}_{s_0,2}$ are considered and the maximum size of these trajectories is unbounded due to the loops $\langle s_0, a_0, s'_1\rangle$ and $\langle s'_1, a_0, s'_1\rangle$.

Precisely, the difference between the look-ahead and short-sighted SSPs is that the former relaxes the original problem by limiting the maximum number of executed actions (horizon), while short-sighted SSPs approximates the original problem by pruning the state space without limiting the problem's horizon. We prove that the optimum value function for short-sighted SSPs is an admissible heuristic for the original SSP and is at least as informative as the look-ahead heuristic (Theorem 2).

Theorem 2. *Given an SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^\mathsf{G}\rangle$, $s \in \mathsf{S}$, $t > 0$ that satisfies the reachability assumption and an admissible heuristic H, then $L_t(s) \leq V^*_{\mathbb{S}_{s,t}}(s) \leq V^*(s)$.*

Proof. By Theorem 1, $\langle V^0_{\mathbb{S}_{s,t}}, \ldots, V^k_{\mathbb{S}_{s,t}}\rangle$ produced by value iteration converges to $V^*_{\mathbb{S}_{s,t}}$ as $k \to \infty$ and is monotonic non-decreasing. By definition of short-sighted SSPs, $V^0_{\mathbb{S}_{s,t}}(s') = V^0(s') = H(s')$ for all $s' \in \mathsf{S}_{s,t}$ and $V^*_{\mathbb{S}_{s,t}}(s) \leq V^*(s)$ since Bellman updates preserve admissibility (Theorem 1). Also, $L_t(s) = V^t(s)$ by definition of t-look-ahead heuristic and the set of states necessary to compute $V^t(s)$ is exactly $\mathsf{S}_{s,t}$. Therefore $L_t(s) = V^t(s) = V^t_{\mathbb{S}_{s,t}}(s) \leq V^*_{\mathbb{S}_{s,t}}(s) \leq V^*(s)$. □

5 Non-Learning Short-Sighted Probabilistic Planner

We present a step towards the definition of our asymptotically optimal planner by describing its basic non-learning version. We also present and prove the guarantees offered by this basic algorithm that are inherited by our main algorithm defined in Section 6. We start by defining a key concept for the rest of the paper:

Definition 4 (t-closed policy). *A policy π is t-closed w.r.t. a state s if, for every trajectory $\langle s, s_{(1)}, \ldots, s_{(k)}\rangle$ generated by π that does not reach a goal state, i.e., $s_{(k)} \notin \mathsf{G}$, then π is not defined for $s_{(k)}$ and $k \geq t$.*

Therefore, t-closed policies guarantee that at least $t - 1$ actions can be executed without replanning. All the replanners reviewed on Section 3 compute 1-closed policies, i.e., they offer no guarantee about the minimum number of actions applied before replanning, and in this section we present an algorithm to compute t-closed policies for arbitrary values of t. Notice that, t-closed policies when $t \to \infty$ are equivalent to closed policies and Theorem 3 gives an upper bound on t for when a t-closed policy becomes a closed policy.

Theorem 3. *Given an SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^\mathsf{G}\rangle$, for all $t \geq |\mathsf{S}|$, every t-closed policy for $\mathbb{S}$ is a closed policy.*

Proof. Suppose $\hat{\pi}$ is a t-closed policy w.r.t. s for $\mathbb{S}$ such that $t \geq |\mathsf{S}|$ and $\hat{\pi}$ is not a closed policy. Since $\hat{\pi}$ is not closed, at least one trajectory $\mathcal{T}_{\hat{\pi}}$ from s finishes because $\hat{\pi}(s_{(k)})$ is not defined and $k \geq t$. By assumption, $t \geq |\mathsf{S}|$, therefore at least one state $s' \in \mathsf{S}$ was visited more than once. Let $\mathcal{T}'$ be a trajectory starting at s and finishing at $s_{(k)}$ such that every state in $\mathcal{T}_{\hat{\pi}}$ is visited at most once. Then, $|\mathcal{T}'| < |\mathcal{T}_{\hat{\pi}}|$ and $\hat{\pi}$ is a t'-closed policy for $t' < t$, a contradiction. □

Besides having partial policies and closed policies as extreme cases, t-closed policies are able to represent the continuum of policies between these two extremes. This characteristic of t-closed policies is key in the relation between SSPs and (s,t)-short-sighted SSPs as shown by the following theorem:

Theorem 4. *Given an SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^\mathsf{G}\rangle$ and a state $s \in \mathsf{S}$ then any closed policy π w.r.t. s for $\mathbb{S}_{s,t}$ is a t-closed policy w.r.t. s for $\mathbb{S}$.*

Proof. Since π is closed for $\mathbb{S}_{s,t}$ w.r.t. s, then every trajectory $\mathcal{T}_\pi$ in $\mathbb{S}_{s,t}$ from s finishes at a state $s' \in \mathsf{G}_{s,t}$. If $s' \notin \mathsf{G}$, then s' is an artificial goal and $|\mathcal{T}_\pi| \geq t$ by the definition of $\mathsf{G}_{s,t}$. Therefore π is t-closed w.r.t. s for $\mathbb{S}$. □

Algorithm 1 shows our algorithm, NoLearning-SSiPP, based on Theorem 4, which generates and solves, on-demand, (s,t)-short-sighted SSPs until a goal state of the original SSP is reached. The procedure SSP-Solver, used by NoLearning-SSiPP, returns a policy that reaches a goal state for the given SSP. To illustrate the execution of NoLearning-SSiPP, consider as input the SSP $\mathbb{S}$ in Figure 1 for $t = 2$. The first short-sighted SSP built by the algorithm is $\mathbb{S}_{s_0,2}$ (Figure 2) and there are only two possible 2-closed policies for it: (i) $\pi_0 = \{(s_0, a_0), (s'_1, a_0)\}$; and (ii)

NOLEARNING-SSIPP(SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^{\mathsf{G}} \rangle$ and $t > 0$)
begin
  $s \leftarrow s_0$
  while $s \notin \mathsf{G}$ **do**
    $\mathbb{S}_{s,t} \leftarrow$ short-sighted-SSP($\mathbb{S}, s, t$)
    $\pi \leftarrow$ SSP-SOLVER($\mathbb{S}_{s,t}$)
    while $s \notin \mathsf{G}_{s,t}$ **do**
      $s \leftarrow$ execute-action($\pi(s)$)

Algorithm 1: Non-learning algorithm to solve SSPs using short-sighted SSPs.

$\pi_1 = \{(s_0, a_1), (s_1, a_1)\}$. If SSP-SOLVER returns π_0 as solution for $\mathbb{S}_{s_0,2}$, then the original SSP $\mathbb{S}$ is solved, since π_0 is a closed policy for $\mathbb{S}$ w.r.t. s_0. Instead, if π_1 is returned by SSP-SOLVER, then $\lceil \frac{k+1}{2} \rceil$ short-sighted SSPs, representing the 3-states subchains of $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_1} \cdots \xrightarrow{a_1} s_{\mathsf{G}}$, are generated and solved by SSP-SOLVER.

6 Learning the Optimal Value Function V^*

We introduce our algorithm, *Short-Sighted Probabilistic Planner* (SSiPP) which is an extension of Algorithm 1 capable of learning the optimal value function V^*. We also prove that SSiPP converges to V^* over the relevant states $\mathsf{S}(\pi^*, s_0)$.

Algorithm 2 presents SSiPP and differs from NOLEARNING-SSIPP in two ways: (i) SSP-SOLVER is replaced by OPTIMAL-SOLVER; and (ii) SSiPP maintains and updates a lower bound $\underline{V}$ for V^*. Given an SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^{\mathsf{G}} \rangle$ and a heuristic H for V^* of $\mathbb{S}$, OPTIMAL-SOLVER returns an optimal policy π^* w.r.t. s_0 for $\mathbb{S}$ and V^* associated to π^*, i.e., V^* needs to be defined only for $s \in \mathsf{S}(\pi^*, s_0)$.

According to the IPPCs (Younes et al. 2005; Bonet and Givan 2007), optimal SSP solvers such as LRTDP are competitive with the best SSP solvers for small to medium problems, therefore restricting SSiPP to optimal solvers should have no practical drawback. Moreover, the usage of an optimal SSP solver brings all the extra guarantees of SSiPP over NOLEARNING-SSIPP, namely the optimality and termination guarantees of SSiPP (Theorem 5 and Corollary 6, respectively).

Theorem 5. *Given an SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^{\mathsf{G}} \rangle$ such that the reachability assumption holds and an admissible heuristic H, then the sequence $\langle \underline{V}_0, \underline{V}_1, \cdots, \underline{V}_t \rangle$, where $\underline{V}_0 = H$ and $\underline{V}_t = SSiPP(\mathbb{S}, t, \underline{V}_{t-1})$, converges to V^* as $t \to \infty$ for all $s \in \mathsf{S}(\pi^*, s_0)$.*

Proof. Let $\mathsf{S}^* \subseteq \mathsf{S}$ be the set of states being visited infinitely many times. Clearly, $\mathsf{S}(\pi^*, s_0) \subseteq \mathsf{S}^*$ since a partial policy cannot be executed ad infinitum without reaching a state in which it is not defined.

In order to show that SSiPP performs Bellman updates implicitly, consider the loop marked with $\diamondsuit$ in Algorithm 2. Since OPTIMAL-SOLVER computes $V^*_{\mathbb{S}_{s,t}}$, by definition of short-sighted SSP: (i) $V^*_{\mathbb{S}_{s,t}}(s_{\mathsf{G}})$ equals $\underline{V}(s_{\mathsf{G}})$ for all $s_{\mathsf{G}} \in \mathsf{G}_{s,t}$, therefore the value of $\underline{V}(s_{\mathsf{G}})$ remains the same; and (ii) $\min_{a \in \mathsf{A}} C(s,a) + \sum_{s' \in \mathsf{S}} P(s'|s,a)\underline{V}(s') \leq V^*_{\mathbb{S}_{s,t}}(s)$ for $s \in \mathsf{S}_{s,t} \setminus \mathsf{G}_{s,t}$, i.e., the assignment $\underline{V}(s) \leftarrow V^*_{\mathbb{S}_{s,t}}$ is equivalent to at least one Bellman update on $\underline{V}(s)$, because $\underline{V}$ is a lower bound on $V^*_{\mathbb{S}_{s,t}}$ and Theorem 1.

SSIPP(SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, C^{\mathsf{G}} \rangle$, $t > 0$ and H a heuristic for V^*)
begin
  $\underline{V} \leftarrow$ Value function for S initialized by H
  $s \leftarrow s_0$
  while $s \notin \mathsf{G}$ **do**
    $\mathbb{S}_{s,t} \leftarrow$ short-sighted-SSP($\mathbb{S}, s, t, \underline{V}$)
    $(\pi^*_{\mathbb{S}_{s,t}}, V^*_{\mathbb{S}_{s,t}}) \leftarrow$ OPTIMAL-SOLVER($\mathbb{S}_{s,t}, \underline{V}$)
    $\diamondsuit$ **forall the** $s \in \mathsf{S}_{s,t}$ **do**
      $\underline{V}(s) \leftarrow V^*_{\mathbb{S}_{s,t}}(s)$
    while $s \notin \mathsf{G}_{s,t}$ **do**
      $s \leftarrow$ execute-action($\pi^*_{\mathbb{S}_{s,t}}(s)$)
  return $\underline{V}$

Algorithm 2: Our asymptotically optimal algorithm to solve SSPs using short-sighted SSPs.

Therefore, we can view the sequence of lower bounds $\langle \underline{V}_0, \underline{V}_1, \cdots, \underline{V}_t \rangle$ generated by SSiPP as asynchronous value iteration. The convergence of $\underline{V}_{t-1}(s)$ to $V^*(s)$ as $t \to \infty$ for all $s \in \mathsf{S}(\pi^*, s_0) \subseteq \mathsf{S}^*$ follows (Bertsekas and Tsitsiklis 1996, Proposition 2.2, p. 27) and guarantees the convergence of SSiPP. □

Corollary 6. *SSiPP always terminates under the conditions of Theorem 5.*

Proof. Suppose SSiPP does not terminate. Then, there is a trajectory $\mathcal{T}$ of infinite size that can be generated by SSiPP. Since S is finite, then there must be an infinite loop in $\mathcal{T}$ and, for all states s in this loop, $\underline{V}(s) \to \infty$ as the execution continues, a contradiction since $\underline{V}(s) \leq V^*(s) < \infty$. □

An interpretation for Theorem 5 is that, given an admissible heuristic H, SSiPP improves the quality of H while maintaining its admissibility and, after enough runs of SSiPP, $\underline{V}$ becomes the perfect heuristic, i.e., V^*. As the experiments presented in the next section show, this combination of replanning and optimality guarantees, not present in any other replanner, translates into improvements in both quality and time to compute the solution of SSPs.

7 Experiments

We present two sets of experiments to: (i) explore the impact of different values of t in the time necessary to compute ϵ-approximations of V^*; and (ii) compare the performance of SSiPP with the winners of the IPPCs 04, 06 and 08 (Younes et al. 2005; Bonet and Givan 2007; Bryce and Buffet 2008).

LRTDP is used as OPTIMAL-SOLVER for SSiPP in all the experiments because it offers an upper bound on the number of iterations necessary to reach ϵ-convergence and it computes closed policies w.r.t. to s_0 instead of closed policies.

All the experiments have the parameter ϵ (for $\epsilon-$convergence) set to 10^{-4} and are conducted in a Linux machine with 4 cores running at 3.07GHz. The planners have a 2Gb memory cut-off and the cpu-time cut-off is different for each set of experiments.

Impact of t in the time to compute V^*

For this set of experiments, we use the race-track domain (Barto, Bradtke, and Singh 1995; Bonet and Geffner 2003). The goal of this domain (Figure 3) is to find a closed policy w.r.t. s_0 for driving a car from the initial state to a set of goal states minimizing the expected cost of the travel. A state is the tuple (x, y, v_x, v_y, b) in which: x, y represent the position of the car in the given 2-D grid (track); v_x, v_y is the velocity in each dimension; and b is true if the car is broken. The car breaks every time it tries to leave the track and the special action FIXCAR is used in order to fix the car (i.e., set b to false) maintaining its position and setting $v_x = v_y = 0$. To move the car, acceleration actions are used. These actions are pairs $(a_x, a_y) \in \{-1, 0, 1\}^2$ denoting the instantaneous acceleration in each direction, which might fail, that is, not change the velocity, with probability 0.1. Acceleration actions have cost 1, while FIXCAR has cost 50.

We consider six race-tracks in this experiment: ring-small, ring-large, square-small, square-large, y-small and y-large. The shape of each track is depicted in Figure 3 and Table 1 presents detailed information about them. For each track, we consider exponentially increasing values of t until $t_{\max}$ is such that $\mathsf{S}(\pi^*, s_0) = \mathsf{S}(\pi^*_{\mathbb{S}_{s_0,t}}, s_0)$, i.e., π^* and the optimal policy for the (s_0, t)-short-sighted SSP are equivalent. Table 1 shows the values of $t_{\max}$ for each track.

In this experiment, we use SSiPP to compute ϵ-approximations of $V^*(s_0)$, i.e., each run consists in executing SSiPP for s_0 multiple times and reusing the computed $\underline{V}$ until $|V^*(s_0) - \underline{V}(s_0)| < \epsilon$. Figure 4 presents the results of this experiment as a log-log plot in which every data point represents the average over 50 runs and the error bars represent the 95% confidence bound of the average run time. The convergence time obtained on $t_{\max}$ can also be obtained for values of t less than $t_{\max}$, showing empirical evidence that there exists a value $t' < t_{\max}$ such that using (t', s)-short-sights SSPs, as opposed to the original SSP, has no effect in the time to compute $V^*(s)$. Such value t' is different for each problem and, for the considered race-tracks, they are: 128 for ring-large and 32 for the other configurations. Figure 4 further shows that, for $t < t'$, the convergence time decreases exponentially as the value of t increases.

Comparison with IPPCs Winners

In this section, we compare SSiPP, LRTDP and the first place planners of the IPPCs over four domains used in IPPC'08, namely blocks world, zeno travel, triangle tire world and exploding blocks world.[2] The first two domains are, respectively, a puzzle and a logistic domain (Veloso 1992). The last two domains, exploding blocks world and triangle tire world, are probabilistically interesting (Little and Thiébaux 2007), i.e., problems in which planners that oversimplify the probabilistic structure of actions have a poor performance. In the exploding blocks world, the extra precondition `(not (= ?b1 ?b2))` in action `put-on-block(?b1 ?b2)` is added to avoid the inconsistent state in which a block is on the top of itself (Little and Thiébaux 2007). For all the problems, we consider that $C(s, a) = 1$ for all $s \in \mathsf{S}$ and $a \in \mathsf{A}$ and $C^{\mathsf{G}}(s) = 0$ for all $s \in \mathsf{G}$.

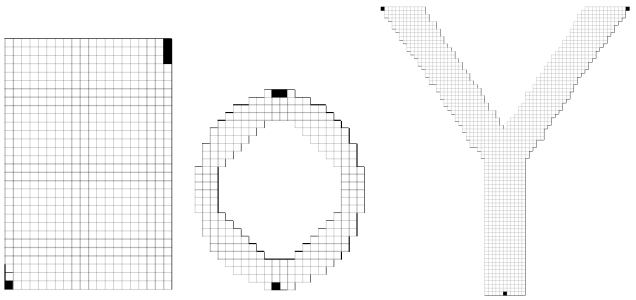

Figure 3: Shape of the race-tracks used. Each cell represents a possible position of the car. The initial state and the goal states are, respectively, the marked cells in the bottom and top of each track.

problem	$\lvert\mathsf{S}\rvert$	% rel.	$t_{\max}$	$V^*(s_0)$	$H_{\mathrm{m}}(s_0)$	t. H_{m}
square-s	42396	2.01	71	18.26	11.00	20.209
square-l	193756	0.75	272	22.26	13.00	145.616
ring-s	4776	12.91	74	21.85	12.00	0.451
ring-l	75364	14.34	869	36.23	24.00	32.056
y-small	101481	10.57	114	29.01	18.00	32.367
y-large	300460	9.42	155	32.81	21.00	211.891

Table 1: Description of each race-track used: size, ratio $\mathsf{S}(\pi^*, s_0)/\mathsf{S}$, $t_{\max}$, $V^*(s_0)$, value of the min-min heuristic (Bonet and Geffner 2003) for s_0 ($H_m(s_0)$) and time in seconds to compute $H_m(s_0)$.

We compare SSiPP against the planners: (i) LRTDP implemented by mGPT (Bonet and Geffner 2005) the 2nd place of IPPC'04; (ii) FF-Replan (winner of IPPC'04); (iii) FPG (winner of IPPC'06) (Buffet and Aberdeen 2009); and (iv) RFF (winner of IPPC'08).[3] We consider 12 different parametrizations of SSiPP obtained by using $t \in \{2, 3, 4, 8\}$ and H as the zero, min-min (Bonet and Geffner 2003) and FF heuristics. The FF heuristic is the only non-admissible heuristic. Given a state s, it equals the length of the solution computed by FF for the all-outcomes determinization of the original problem using s as initial state. For LRTDP, we consider the same 12 parametrizations as in SSiPP, where t is the number of look-ahead steps, and LRTDP without look-ahead (i.e., $t = 1$), totalizing 15 different configurations.

We use a methodology similar to the one in IPPC'04 and IPPC'06, in which there is a time cutoff for each individ-

[2]The other domains of IPPC'08 and the problems of IPPC'11 are not considered because our current implementation of SSiPP does not fully support PPDDL.

[3]The planner Prost, winner of IPPC'11, is not currently available.

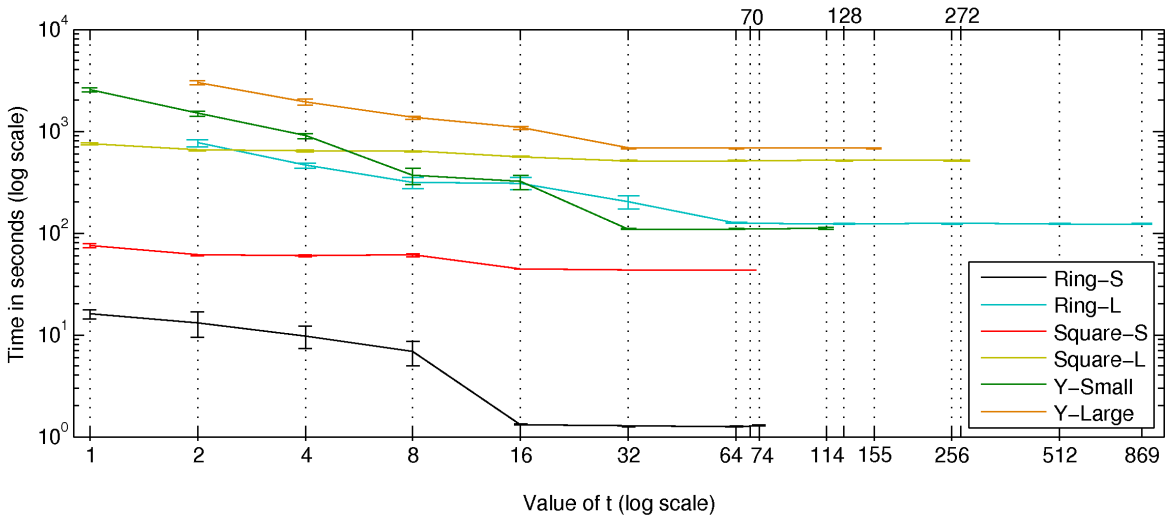

Figure 4: Time to compute ϵ-approximations of $V^*(s_0)$ versus short-sighted size t. The last point in the line for each track, $t_{\max}$, is such that $\mathsf{S}(\pi^*, s_0) = \mathsf{S}(\pi^*_{\mathbb{S}_{s_0,t_{\max}}}, s_0)$, therefore the performance is the same for $t \geq t_{\max}$. The error bars represent the 95% confidence interval for the average run time.

ual problem. Formally, each problem needs to be solved 50 times in 20 minutes using the MDPSIM (Younes et al. 2005), an environment simulator. The computation of each solution for the same problem is called a round and training is allowed between rounds, i.e., the knowledge obtained from one round, e.g., the lower bound on $V^*(s_0)$, can be used to solve subsequent rounds. The evaluation is done by the number of rounds simulated by MDPSIM that reached a goal state. The maximum number of actions allowed per round is 2000 and rounds that exceed this limit are stopped by MDPSIM and declared as failure, i.e., goal not reached.

In order to respect the time cutoff, SSiPP simulates rounds internally until it ϵ-converged or the remaining time to solve the problem is smaller than $50 \times (avg + 2.33 \times sem)$, where avg is the average time to internally solve a round of the given problem and sem is the standard error of the mean. This deadline is an approximation the upper part of the 98% confidence interval for the time to simulate 50 rounds. When the deadline is reached, SSiPP starts the 50 rounds simulations through MDPSIM. We employ the same approach for LRTDP using the LRTDP-TRIAL function (Bonet and Geffner 2003).

Figure 5 shows the results on parametrizations of SSiPP: SSiPP-O, the overall best configuration; and SSiPP-D, the best configuration for each domain. The parametrization of SSiPP-O is $t = 3$ and the FF heuristic; the parametrization of SSiPP-D is: $t = 8$ and the zero heuristic for the triangle tire world; $t = 3$ and the FF heuristic (same as in SSiPP-O) for the exploding blocks world; and $t = 2$ and the FF heuristic for both the zeno travel domain and the blocks world. For LRTDP, only the best configuration for each domain is presented (LRTDP-D) and these parametrizations are the same as in SSiPP-D except for the triangle tire world, in which LRTDP-D corresponds to $t = 3$ and zero heuristic. This difference between SSiPP-D and LRTDP-D is because LRTDP using t-look-ahead does not scale up for $t > 3$ in the triangle tire world. Also, the zero heuristic obtains the best performance for the triangle tire domain because both the FF heuristic and the min-min heuristics favor the shortest path from the initial state to the goal state and the probability of this path reaching a dead-end is positive and increases exponentially with the problem size (Little and Thiébaux 2007).

SSiPP-D perform at least as good as FF-Replan and RFF in, respectively, 81.7% and 86.7% of the problems and never performs worse than LRTDP-D and FPG. The performance of SSiPP-D is especially noteworthy in the triangle tire world in which it scales up better than the other planners and outperforms them in the 7 largest instances. SSiPP-O perform at least as good as RFF, FF-Replan and LRTDP-D in, respectively, 73.3%, 81.7% and 85% of the problems and never worse than FPG.

8 Conclusion

In this paper we introduced short-sighted SSPs, a model for probabilistic planning problems that offers a new approach for relaxing SSPs. This approach consists of pruning the state space based on the non-symmetric distance between states and defining artificial goal states that guide the solution towards the original goals. We proved that closed policies to (s, t)-short-sighted SSPs can be executed in the original problem for at least t actions starting from s. Moreover, we also proved that the optimal value function for an

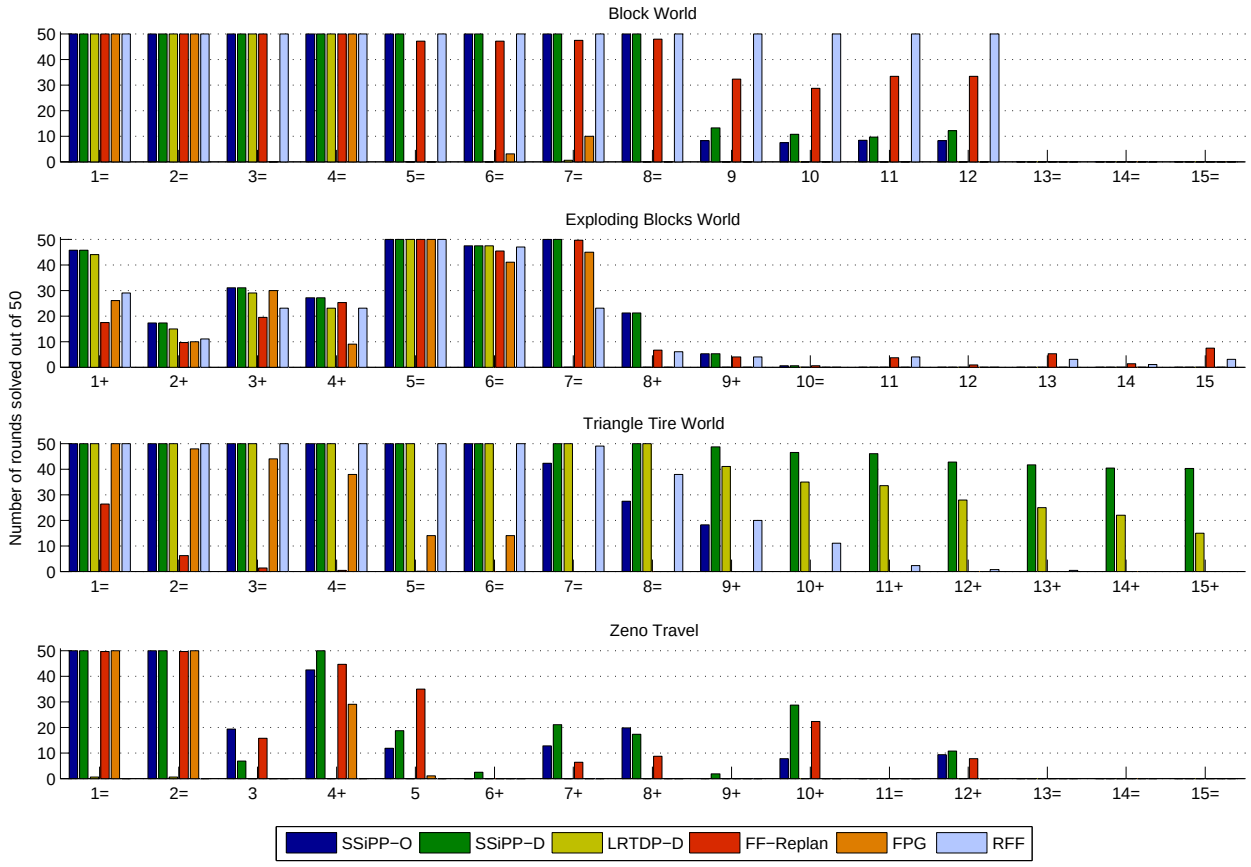

Figure 5: Number of rounds solved for each problem in the domains considered. Only 2 parametrizations of SSiPP are shown: SSiPP-D, the best parametrization for the given domain; and SSiPP-O, the best setting when considering all the problems. In the number of the problems (x-axis), the symbols $+$ and $=$ represent, respectively, problems in which SSiPP-D outperforms the other planners and ties with the best competitor.

(s, t)-short-sighted SSPs is a lower bound, i.e., an admissible heuristic, for the optimal value function of the original SSP and is more informative than the look-ahead heuristic.

We introduced two algorithms to solve SSPs using short-sighted SSPs, namely NoLearning-SSiPP and SSiPP. Both algorithms guarantee that at least t actions are executed before replanning and, for $t \geq t_{\max}$, replanning is not needed. Therefore, by varying t, NoLearning-SSiPP and SSiPP can behave as either probabilistic planners by computing closed policies, or replanners by computing partial policies. On the theoretical side, we bounded the value of $t_{\max}$ and proved that SSiPP is asymptotically optimal, making SSiPP the only planner that, at the same time, guarantees optimality and offers a bound in the minimum number of actions executed without replanning. On the empirical side, we found a correlation in the value of t and the time to compute ϵ-approximations of the optimal solution. This trend suggests that, for different problems, there exists $t' < t_{\max}$ such that the time to converge to the optimal solution remains approximately the same for $t \geq t'$. We also empirically compared SSiPP with LRTDP and the available winners of the probabilistic planning competitions and, in 81.7% of the problems, SSiPP performs at least as good as the best competitor.

Our current research agenda includes: (i) adding full support of PPDDL to SSiPP; (ii) further explore the relation between t', $t_{\max}$ and domain metrics; and (iii) combine the replanning guarantees of SSiPP and RFF.

Acknowledgments

We would like to thank Marcelo Hashimoto, Charalampos Tsourakakis and the anonymous reviewers for their insightful comments and Blai Bonet, Hector Geffner (mGPT), Olivier Buffet, Douglas Aberdeen (FPG), Florent Teichteil-Konigsbuch, Guillaume Infantes and Ugur Kuter (RFF) for making the code of their planners available.

References

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72(1-2):81–138.

Bertsekas, D., and Tsitsiklis, J. 1991. An analysis of stochastic shortest path problems. *Mathematics of Operations Research* 16(3):580–595.

Bertsekas, D., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Bertsekas, D. 1995. *Dynamic Programming and Optimal Control*. Athena Scientific.

Bonet, B., and Geffner, H. 2003. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*.

Bonet, B., and Geffner, H. 2005. mGPT: A probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research* 24.

Bonet, B., and Givan, R. 2007. 2th International Probabilistic Planning Competition (IPPC-ICAPS'06). http://www.ldc.usb.ve/~bonet/ipc5/ (accessed on Dec 13, 2011).

Bryce, D., and Buffet, O. 2008. 6th International Planning Competition: Uncertainty Track. In *3rd International Probabilistic Planning Competition (IPPC-ICAPS'08)*.

Buffet, O., and Aberdeen, D. 2009. The factored policy-gradient planner. *Artificial Intelligence* 173(5-6):722–747.

Dean, T.; Kaelbling, L.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76(1-2):35–74.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14(1):253–302.

Kocsis, L., and Szepesvri, C. 2006. Bandit based Monte-Carlo Planning. In *Proceedings of the European Conference on Machine Learning (ECML'06)*.

Kolobov, A.; Mausam; and Weld, D. S. 2009. ReTrASE: Integrating Paradigms for Approximate Probabilistic Planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*.

Little, I., and Thiébaux, S. 2007. Probabilistic planning vs replanning. In *Proceedings of ICAPS Workshop on IPC: Past, Present and Future*.

McMahan, H.; Likhachev, M.; and Gordon, G. 2005. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*.

Pearl, J. 1985. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Menlo Park, California: Addison-Wesley.

Puterman, M. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Smith, T., and Simmons, R. G. 2006. Focused Real-Time Dynamic Programming for MDPs: Squeezing More Out of a Heuristic. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*.

Teichteil-Koenigsbuch, F.; Infantes, G.; and Kuter, U. 2008. RFF: A robust, FF-based mdp planning algorithm for generating policies with low probability of failure. *3rd International Planning Competition (IPPC-ICAPS'08)*.

Veloso, M. 1992. *Learning by analogical reasoning in general problem solving*. Ph.D. Dissertation, Carnegie Mellon University.

Yoon, S.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic planning via determinization in hindsight. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*.

Yoon, S.; Ruml, W.; Benton, J.; and Do, M. B. 2010. Improving Determinization in Hindsight for Online Probabilistic Planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*.

Younes, H.; Littman, M.; Weissman, D.; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research* 24(1):851–887.

Zickler, S., and Veloso, M. 2010. Variable Level-Of-Detail Motion Planning in Environments with Poorly Predictable Bodies. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*.