Soft Labeling in Stochastic Shortest Path Problems

Luis Pineda[1,2] and Shlomo Zilberstein[1]
[1]University of Massachusetts, Amherst, MA
[2]Facebook AI Research, Montreal, QC
lpineda,shlomo@cs.umass.edu

ABSTRACT

The Stochastic Shortest Path (SSP) is an established model for goal-directed probabilistic planning. Despite its broad applicability, wide adoption of the model has been impaired by its high computational complexity. Efforts to address this challenge have produced promising algorithms that leverage two popular mechanisms: *labeling* and *short-sightedness*. The resulting algorithms can generate near-optimal solutions much faster than optimal solvers, albeit at the cost of poor theoretical guarantees. In this work, we introduce a generalization of labeling, called *soft labeling*, which results in a framework that encompasses a wide spectrum of efficient labeling algorithms, and offers better theoretical guarantees than existing short-sighted labeling approaches. We also propose a novel instantiation of this framework, the SOFT-FLARES algorithm, which achieves state-of-the-art performance on a diverse set of benchmarks.

KEYWORDS

Markov decision processes; probabilistic planning; short-sighted algorithms; stochastic shortest path problems

ACM Reference Format:
Luis Pineda and Shlomo Zilberstein. 2019. Soft Labeling in Stochastic Shortest Path Problems. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

The goal of this work is to provide a better understanding and more effective algorithms for solving Stochastic Shortest Path (SSP) problems [2]. SSPs are a class of goal-directed Markov Decision Process [17], where the objective is to minimize the expected cost of reaching a goal state from a fixed initial state. Many practical applications involving autonomous agents can be modeled as SSPs, such as decision systems for fighting wildfires [8], planning in semi-autonomous vehicles [24], and automated charging of electrical vehicles [9]. Unfortunately, the complexity of solving large SSPs [14] has impaired the widespread adoption of the model. Developing effective SSP solvers remains a challenging research problem.

Two techniques have contributed tremendously to progress in the area of SSP solvers: *labeling* and *short-sightedness*. Labeling techniques identify states for which further computation will not improve value estimates (a proxy for policy quality) above a given tolerance. For MDPs and SSPs, the technique was first introduced by Bonet and Geffner [4]. Their LRTDP algorithm included a labeling procedure that, starting from some given state, visits all states that can be reached using the current policy; if the values of all of these states have converged up to some tolerance, the algorithm labels all of them as "solved". This technique accelerates computation significantly, by allowing a solver to avoid states for which computation is no longer productive.

The other influential development, short-sightedness, is a popular technique for solving planning problems *approximately*, by limiting the search to states that are "close" to the initial state. This can substantially speed up computation in exchange for some degradation in solution quality. In the case of SSPs, the most prominent short-sighted algorithms are HDP [3] and SSIPP [23]. However, the disadvantages of typical short-sighted methods are a strong dependence on having a good heuristic and possibly requiring large horizons to produce plans of good quality.

Recent work by Pineda *et al.* [16] combined these two ideas by exploring the use of short-sightedness only during the labeling phase, and otherwise allowing the planner to explore the state space freely. Intuitively, this allows the solver to explore deeper into the relevant parts of the state space, while still resulting in substantial computational savings. An algorithm based on this idea, called FLARES, was shown empirically to have advantages over LRTDP, HDP and SSIPP, producing near-optimal plans orders of magnitude faster than LRTDP. However, despite showing promising empirical results, FLARES does not possess any theoretical guarantees such as optimality or ϵ-consistency—even if allowed infinite computation time—unless the la beling horizon is chosen appropriately. Unfortunately, there is no known principle for choosing the necessary labeling horizon, other than trial and error or prior domain knowledge. Conversely, standard—non-myopic—labeling can be used to provide theoretical guarantees, with the potential downside of requiring the exploration of a very large number of states.

To bridge the gap between these two extremes, the main contribution of this work is a general framework, *soft labeling*, for describing short-sighted labeling-based SSPs solvers. Soft labeling generalizes the purpose of labeling, allowing it to modify the sampling strategy, by reducing the probability of sampling states that are close to convergence, in a topological sense. We propose, and build upon, a new measure that we call *ϵ-distance*, which represents how close a state is to states with large residual errors. We show how to leverage this quantity to direct the search to parts of the state space where computation would be more productive. Furthermore, we present an algorithm based on these ideas, SOFT-FLARES, that achieves the computational efficiency of FLARES, and can also provide stronger theoretical guarantees. Our experiments show that SOFT-FLARES is competitive with state-of-the-art SSP solvers, both in terms of computation time and policy performance. We also analyze some of the drawbacks and potential improvements of our framework, as well as directions on how to overcome these existing limitations.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 RELATED WORK

The use of labeling for solving MDPs was first introduced by Bonet and Geffner [4] through the LRTDP algorithm; the technique has been further explored in algorithms such as HDP [3], LSDF [6], LR^{2}TDP [12], and, more recently, FLARES [16]. Among these algorithms, this work is closest to HDP(I) and FLARES. As in our proposed framework, the former combines labeling and short-sightedness, by searching over states that can be reached with high probability from the initial state (or the current state of execution, when an online planning strategy is used). However, an algorithm using soft labeling, which we show FLARES is an instance of, is allowed to freely explore the state space, and the search is only constrained whenever the algorithm attempts to label a state as solved. In this respect, our work generalizes the notion of short-sighted labeling introduced by FLARES, and we propose a framework that encompasses a wide spectrum of labeling-based algorithms filling the gap between FLARES and LRTDP.

Additionally, short-sightedness in the solution of SSPs has also been explored by Trevizan and Veloso, with the introduction of the SSIPP algorithm [23]. Multiple variants of this algorithm exist, particularly trajectory-based SSIPP [21], SSIPP-FF, and LABELED-SSIPP [23]. As in the case of HDP(I), all of these variants restrict the search to states close to the initial state, thus potentially requiring large horizons to obtain good performance.

Finally, our work also bears similarity to a set of extensions of RTDP that exploit upper bounds on state values to direct exploration: BRTDP [15], FRTDP [19], and VPI-RTDP [18]. The main similarity between these algorithms and our soft labeling framework is in the introduction of a bias in the transition function used for sampling, so that exploration is directed to parts of the state space that are further away from convergence. However, unlike BRTDP, FRTDP and VPI-RTDP, our framework does not require initial upper bounds for state values. While this may look like a minor improvement, note that there are currently no practical methods to efficiently compute good upper bounds for state values in SSPs. For instance, the method proposed in the BRTDP paper requires a pass through every state in the problem domain, a computational cost that can be prohibitive in large SSPs with thousands, millions, or even billions of states. Without an efficient automatic approach to find upper bounds, the user is left with no choice but to provide some problem-dependent initial values, possibly requiring specialized domain knowledge. On the other hand, our framework only requires the user to provide an initial heuristic (i.e., a lower bound initialization of state values), which is standard among state-of-the-art SSP solvers.

3 FORMAL BACKGROUND

We primarily focus on Stochastic Shortest Path problems (SSPs) [2]. An SSP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, C, s_0, \mathcal{G} \rangle$, where: $\mathcal{S}$ is the finite set of all possible *states* of the system; $\mathcal{A}$ is the finite set of all possible *actions* the agent can take; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is a *transition function* specifying the probability $\mathcal{T}(s,a,s')$ of outcome state s' whenever action a is executed in state s; $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a *cost function* that gives the cost $C(s,a)$ incurred whenever the agent executes action a and the system is in state s; $s_0 \in \mathcal{S}$ is the *initial state* of the system; and $\mathcal{G} \subseteq \mathcal{S}$ is the non-empty set of *goal states*, s.t. for every $s_g \in \mathcal{G}$, for all $a \in \mathcal{A}$, and for all $s' \neq s_g$, the transition function obeys $\mathcal{T}(s_g, a, s_g) = 1$, $\mathcal{T}(s_g, a, s') = 0$, and $C(s_g, a, s_g) = 0$.

The objective in an SSP is to bring the system from the initial state s_0 to a goal state $s_g \in \mathcal{G}$ with minimum total cost, in expectation. The behavior of an agent is described in terms of a *policy*, $\pi : \mathcal{S} \to \mathcal{A}$, a mapping that assigns an action to every state. Given a policy π, we can define the *value function* V^π that represents the expected total cost incurred when π is executed starting from state s. That is,

$$V^\pi(s) \triangleq \mathbb{E}\Big[\sum_{t=0}^{\infty} C(s_t, \pi(s_t)) | s_0 = s, \pi\Big] \qquad (1)$$

A policy π is called *proper* if following π from any given state has a probability 1 of reaching a goal; otherwise, it is called *improper*. An optimal solution to an SSP, or *optimal policy*, denoted as π^*, and its *optimal value function*, V^*, are ones that satisfy,

$$V^*(s) = \min_\pi V^\pi(s) \qquad (2)$$

$$V^*(s) = \min_a \Big[C(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V^*(s')\Big] \qquad (3)$$

$$\pi^*(s) = \arg\min_a \Big[C(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V^*(s')\Big] \qquad (4)$$

as long as the following two conditions are met [2]: i) there exists at least one proper policy, and ii) for every improper policy π, and for every state $s \in \mathcal{S}$ where π is improper, $V^\pi(s) = \infty$. Under these two conditions, the optimal value function for an SSP, V^*, is the fixed point of the set of *Bellman equations*,

$$V(s) = \min_a \Big[C(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V(s')\Big] \qquad (5)$$

Applying (5) to a state s is commonly referred to as a *Bellman backup*. It is also useful to introduce the concept of *residual* at a state, defined as

$$Res^V(s) \triangleq \Big| V(s) - \min_{a \in \mathcal{A}} \Big[C(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V(s')\Big]\Big|$$

Finally, a state s is called ϵ-consistent if $Res^V(s) < \epsilon$.

4 SOFT LABELING IN SSPS

Labeling in MDPs refers to a mechanism for caching the value of states that are guaranteed to be ϵ-consistent. This property can be established by confirming that all the states reachable from a given state, under the current greedy policy, are also ϵ-consistent [4]. If so, all of the reachable states can be labeled as solved. This approach often produces substantial computational savings, allowing search algorithms to avoid working on states whose value have converged within an acceptable tolerance.

4.1 Generalizing Labeling

Labeling in SSPs can be interpreted as an outcome selection mechanism [10] that continually modifies the transition function used for sampling states during planning. Specifically, it modifies the probabilities of sampling successor states guaranteed to remain ϵ-consistent, making these probabilities equal to zero. In this work we introduce *soft labeling*, which generalizes labeling as a probabilistic factor that modifies the transition function used for sampling. We begin formalizing the notion of soft labeling by introducing a few key definitions.

Definition 4.1. **Deterministic policy graph rooted at a state**. Given an SSP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, C, s_0, \mathcal{G} \rangle$ and a state $s \in \mathcal{S}$, the deterministic policy graph rooted at state s is a directed graph $G_{s,\pi} = (\mathcal{S}_{s,\pi}, E_\pi)$, where the set of vertices, $\mathcal{S}_{s,\pi}$, is the set of all states reachable from s by following policy π, and E_π is a set of edges $\{\langle s', s'' \rangle \mid \mathcal{T}(s', \pi(s'), s'') > 0\}$.

That is, $G_{s,\pi}$ is a digraph containing a vertex for every state reachable from s following policy π, and an edge connecting two states whenever one is a possible outcome of the other under π.

Definition 4.2. **Weighted distance between two states**. Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, C, s_0, \mathcal{G} \rangle$ be an SSP and s a state in $\mathcal{S}$. Furthermore, let $G^{(w)}_{s,\pi}$ be the deterministic policy graph, weighted with a function $w : E^\pi \longrightarrow \mathbb{R}^+_0$ that assigns a non-negative weight to each edge. Then, the weighted distance between s and $s' \in \mathcal{S} \setminus \{s\}$, $\delta(s, s')$ is the total weight of the shortest path between s and s' in the weighted deterministic policy graph. When $s = s'$, $\delta(s, s') \triangleq 0$.

This notion of weighted distance allows us to characterize different measures of short-sightedness using a single notation. For example, the most common—depth-based—form, which considers the minimum number of actions needed to reach another state, can be represented by assigning $w(\langle s, s' \rangle) = 1$ to every edge in $G^{(w)}_{s,\pi}$. Additionally, we can represent other forms of short-sightedness based on trajectory probabilities [23], using

$$w(\langle s, s' \rangle) = -\log_2 \mathcal{T}(s, \pi(s), s') \tag{6}$$

or plausibilities [3], using

$$w(\langle s, s' \rangle) = \left\lfloor -\log_2 \left(\frac{\mathcal{T}(s, \pi(s), s')}{\max_{s''} \mathcal{T}(s, \pi(s), s'')} \right) \right\rfloor \tag{7}$$

Given a weight function, w, we define the ϵ-distance of state s, $\eth_\epsilon(s)$, as the shortest weighted distance from s to a state that is not ϵ-consistent.

Definition 4.3. **ϵ-distance of a state**. Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, C, s_0, \mathcal{G} \rangle$ be an SSP, s a state in $\mathcal{S}$, and $G^{(w)}_{s,\pi}$ be the weighted deterministic policy graph. Furthermore, let $V : \mathcal{S} \longrightarrow \mathbb{R}$ be a value function. The ϵ-distance of state s, $\eth_\epsilon(s)$, is defined as $\eth_\epsilon \triangleq \min_{s' \in \mathcal{S}^{\epsilon+}_s} \delta(s, s')$, where $\mathcal{S}^{\epsilon+}_s \triangleq \{s' \in \mathcal{S}_{s,\pi} \mid \text{Res}^V(s) > \epsilon\}$, and $\eth_\epsilon = \infty$ if $\mathcal{S}^{\epsilon+}_s = \emptyset$.

Note that $\eth_\epsilon(s)$ generally depends on the weight function as well as the current policy and value function. For the sake of clarity, we omit these details from the notation, but we make sure that in the rest of the paper the correct conditions are clear from the context. Figure 1 illustrates the ϵ-distance of a few states on a small SSP, using a depth-based weight function. Assuming $\epsilon = 0.1$, the red edges illustrates the path to the state at the shortest weighted distance from A (G), using $w(\langle s, s' \rangle) = 1$, which results in $\eth_\epsilon(A) = 2$. The blue path shows the corresponding path (to state H) when w is defined as in (6), resulting in $\eth_\epsilon(A) = 0.46$.

We can use the concept of ϵ-distance to provide a concise definition of the typical criterion for labeling states in SSPs solvers [4]: *a state s should be labeled only if $\eth_\epsilon(s) = \infty$ under the current value function V and a greedy policy over V (irrespective of the weight function used).* Additionally, the depth-based short-sighted labeling criterion [16] can be described as: *a state s should be labeled only if $\eth_\epsilon(s) \geq t$, where t is an input parameter and the ϵ-distance is*

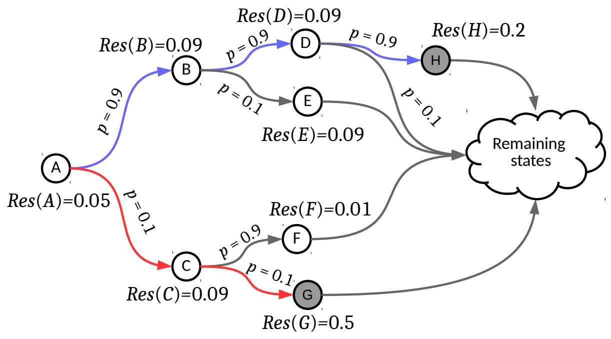

Figure 1: Illustration of the ϵ-distance of a state under a given policy, for two different distance functions.

conditioned on the current value function V, a greedy policy π over V, and $w(e) = 1$ for every edge in the deterministic policy graph.

Both of these labeling criteria consider states solved once they are labeled. In practice, this means that trial-based algorithms using labeling (e.g., LRTDP or FLARES) stop trials as soon as they encounter labeled states. We can describe this process via a *sampling function*, $\sigma : \mathcal{S} \times \mathcal{A} \times \mathcal{S}^+ \longrightarrow [0, 1]$, such that $\sigma(s, a, s')$ represents the probability that a trial continues in state s' if action a is chosen when visiting state s. We use the notation $\mathcal{S}^+ \triangleq \mathcal{S} \cup \{\hat{s}\}$, where $\hat{s}$ is a dummy state that represents that the current trial stops. Note that σ only affects the algorithm's choice of explored states, not the computation of values using (5).

Building on this notation, we can represent labeling-based sampling via

$$\sigma(s, a, s') = \begin{cases} \left(1 - \mathcal{L}(s')\right) \cdot \mathcal{T}(s, a, s') & \text{if } s' \in \mathcal{S} \\ \sum_{x \in \mathcal{S}} \mathcal{L}(x) \cdot \mathcal{T}(s, a, x) & \text{if } s' = \hat{s} \end{cases} \tag{8}$$

where the label, $\mathcal{L}$, is a factor that alters the probability of a trial continuing at a given successor state.

This definition of labeling generalizes existing forms of labeling, which can be recovered from (8) with appropriate definitions of $\mathcal{L}$. For example, to recover the labeling used in LRTDP, $\mathcal{L}$ should be defined as

$$\mathcal{L}(s') \triangleq [\eth_\epsilon(s') = \infty] \tag{9}$$

where $\eth_\epsilon(s')$ is conditioned on V, the greedy policy over V, and an arbitrary weighting function w; $[\cdot]$ denotes an Iverson bracket. For the depth-based short-sighted labeling criterion used in FLARES, the label is

$$\mathcal{L}(s') \triangleq [\forall s'' \in \mathcal{S}_{s',\pi} \cap \{s'' : \delta(\langle s', s'' \rangle) \leq t)\}, \ \eth_\epsilon(s'') \geq t] \tag{10}$$

where $\eth_\epsilon(s'')$ is conditioned on V, its associated greedy policy π, and the weight function $w(\langle s, s' \rangle) = 1$. While this labeling function might seem overly complicated, in later sections we show that we can generalize this behavior by means of a generic procedure for estimating ϵ-distances, while letting the labeling functions be dependent only on the estimated value of $\eth_\epsilon(s')$.

4.2 Soft Labeling

Algorithm 1 presents a generic trial-based solver based on the soft labeling framework described above. The algorithm receives a labeling function, $\mathcal{L}$, a weight function used to compute distances, w, the residual tolerance to be used, ϵ, the number of trials to perform, n, and a vector with additional parameters, θ (e.g., the horizon t in FLARES).

The algorithm starts by initializing ϵ-distances of all states (line 1)[1]. Typically, ϵ-distances should be initialized so that $\mathcal{L}(s) = 0$ (e.g., by setting $\mathfrak{d}_\epsilon(s)$ to $-\infty$), but we allow room for other possibilities, such as keeping ϵ-distances computed during previous calls to the solver on the same input problem. The algorithm functions in a manner very similar to LRTDP, with the following differences:

- The label s.SOLVED is replaced by a random sample $[x \sim Bernoulli(\mathcal{L}(s)) = 1]$ (lines 13 and 18), which is consistent with the probabilistic interpretation of $\mathcal{L}$ and will become more relevant when we introduce soft versions of $\mathcal{L}$.
- The states sampled during the trials (line 12) are sampled according to (8) (function SAMPLE-FROM-SIGMA).
- The call to CHECK-SOLVED(s) is replaced with a call to function ESTIMATE-ϵ-distance(s) (line 17) . The objective of this function is to explore states in $\mathcal{S}_{s,\pi}$, where π is the greedy policy on the current value estimates, and estimate ϵ-distances for s (and possibly for other states in the graph). The function receives the distance function to be used, w, and any additional parameters necessary, θ.

This generic SOFT-LRTDP algorithm generalizes LRTDP and FLARES, as long as ESTIMATE-ϵ-distance(s) is instantiated appropriately. For example, for obtaining LRTDP, it needs to explore all states in $s' \in S_{s,\pi}$, and set $\mathfrak{d}_\epsilon(s') \leftarrow \infty$ iff $\text{Res}^V(s') \leq \epsilon$ for all s'. For obtaining FLARES, we can implement ESTIMATE-ϵ-distance as a short-sighted version of CHECK-SOLVED(s) that: i) limits the search to depth $2t$, and ii) sets $\mathfrak{d}_\epsilon(s') = t$ for any s' found up to depth t iff all states explored satisfy $\text{Res}^V(s') \leq \epsilon$. Moreover, our framework allows extensions of FLARES that use other distance measures, such as trajectory probabilities or plausibilities.

Although reformulating existing algorithms in this light is somewhat interesting, it does not immediately result in drastically different solution methods for SSPs. However, as we show next, the real power of this framework is that it directly implies a family of labeling mechanisms that achieve the computational efficiency of FLARES, while still maintaining theoretical guarantees of performance. The main insight is to realize that there is nothing forcing us to use an indicator function for $\mathcal{L}$; in fact, we can use any arbitrary function $\mathcal{L} : \mathcal{S} \rightarrow [0, 1]$. We refer to the resulting outcome selection approach as *soft labeling* because it allows the labeling function, $\mathcal{L}$, to deter—but not prevent—a state from being explored. Furthermore, as we show in our experiments, the flexibility in the choice of labeling function can also be leveraged when using approximate solvers, by biasing the search towards states where computation can be more productive, resulting in faster planning without significant impact on policy quality.

4.3 Theoretical Properties

The use of soft labeling leads to theoretical properties that cannot be obtained with deterministic short-sighted labels, such as

[1]In practice this should be done lazily, i.e., whenever a state's ϵ-distance needs to be used the first time. We explicitly include it here to highlight the prominent role ϵ-distances play in the algorithm.

Algorithm 1: A generic soft labeling trial-based algorithm based on RTDP.

```
SOFT-LABELED-RTDP
    input   : $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, C, s_0, \mathcal{G} \rangle$, $\mathcal{L}$, $w$, $\epsilon$, $n$, $\theta$
    output  : an action to execute
1   $\forall s \in \mathcal{S}$, $\mathfrak{d}_\epsilon(s) \leftarrow$ INITIALIZE-$\epsilon$-DISTANCE($s$)
2   $i \leftarrow 0$
3   while $i < n$ do
4       $i \leftarrow i + 1$
5       $s = s_0$
6       $visited \leftarrow$ EMPTY-STACK
7       while true do
8           $visited$.PUSH($s$)
9           if $s \in \mathcal{G}$ then break
10          BELLMAN-UPDATE($s$)
11          $a \leftarrow$ GREEDY-ACTION($s$)
12          $s \leftarrow$ SAMPLE-FROM-SIGMA($s$, $a$, $\mathcal{T}$, $\mathcal{L}$, $\mathfrak{d}$)
13          if $[x \sim Bernoulli(\mathcal{L}(s)) = 1]$ then
14              break
15      while $visited \neq$ EMPTY-STACK do
16          $s \leftarrow visited$.POP()
17          $\mathfrak{d} \leftarrow$ ESTIMATE-$\epsilon$-distance($\mathcal{M}$, $s$, $w$, $\theta$)
18          if $[x \sim Bernoulli(\mathcal{L}(s)) = 0]$ then
19              break
20  return GREEDY-ACTION($s_0$)
```

those used by FLARES. In particular, Theorem 4.4 below shows conditions under which SOFT-LRTDP can produce optimal policies, by operating similarly to RTDP. Further, Theorem 4.5 shows conditions on ESTIMATE-ϵ-distance under which the algorithm converges to ϵ-consistent values with high probability, thus operating similarly to LRTDP. Note that, crucially, the use of ψ in Theorem 4.5 implies the existence of a wide spectrum of short-sighted labeling strategies that allow SOFT-LRTDP to bridge the gap between RTDP and LRTDP.

THEOREM 4.4. *Given, i) a labeling function $\mathcal{L}$ such that $\forall s' \in \mathcal{S}, \mathcal{L}(s') < \eta < 1$, for some fixed η, and ii) an implementation of ESTIMATE-ϵ-distance(s) that only changes state values through Bellman backups, and iii) an admissible initial value function, then repeated trials of Algorithm 1 eventually yield optimal values over all states reachable by a greedy policy on the states values.*

PROOF. This follows from the optimality of asynchronous value iteration and RTDP [1]. Conditions i-iii) ensure that SOFT-LRTDP operates like RTDP, with the only difference being the sampling probabilities used during the trials. Restricting $\mathcal{L}(s') < \eta < 1$ guarantees that repeated trials can visit states in any optimal policy infinitely often. □

THEOREM 4.5. *Consider, i) a labeling function $\mathcal{L}$ such that $\mathcal{L}(s') = 1$ iff $\mathfrak{d}_\epsilon(s') = \infty$; ii) an implementation of function ESTIMATE-ϵ-distance(s) that sets $\mathfrak{d}_\epsilon(s') = \infty$ iff $\mathcal{S}_{s'}^{\epsilon+} = \emptyset$, only changes state values through Bellman backups, and with probability $\psi > 0$ it explores all states in $\mathcal{S}_{s,\pi}$; and iii) an admissible initial value function. Then, under conditions i-iii) and for any $0 < p < 1$, there exists a value $N_p > 0$ s.t. the probability that N_p trials of Algorithm 1 yield*

ϵ-consistent values over all states reachable by the greedy policy is higher than p.

Proof. Under conditions i-iii) there is a probability ψ that a call to ESTIMATE-ϵ-distance operates exactly like the CHECK-SOLVED function of LRTDP. Suppose SOFT-FLARES never terminates under these conditions when $n \longrightarrow \infty$. Then, there must be a state s_i such that ESTIMATE-ϵ-distance(s_i) is called an infinite number of times. Also note that, following Bonet and Geffner [4], there is a finite number of calls to CHECK-SOLVED(s_i) after which $\mathcal{S}^{\epsilon+}_{s_i} = \emptyset$. This maximum number of calls to CHECK-SOLVED(s_i) is bounded by

$$C = \epsilon^{-1} \sum_{s \in S} V^*(s) - h(s) \qquad (11)$$

where $h(s)$ is the initial admissible value function. Because $\psi > 0$, we can then bound the probability that after n calls to function ESTIMATE-ϵ-distance(s_i), $\mathcal{S}^{\epsilon+}_{s_i} \neq \emptyset$. Let X be the random variable representing the total number of calls to ESTIMATE-ϵ-distance(s_i) that are equivalent to CHECK-SOLVED(s_i). Then, by applying Chernoff bound,

$$Pr(X \leq C) \leq \exp\left\{ -(n\psi - C)^2 / 2n\psi \right\} \qquad (12)$$

Thus, for any $0 < q < 1$, as long as $N_q - \sqrt{2N_q\psi \log 1/q} > C$, then N_q calls ensure $Pr(X \leq C) < q$. Together with conditions i-ii), this implies that after N_q calls to ESTIMATE-ϵ-distance, s_i will be labeled $\mathcal{L}(s_i) = 1$ with probability higher than q. Moreover, since q can be arbitrarily small and the number of states is finite, this implies that for any probability $p < q$ there is a number of SOFT-FLARES trials, N_p, after which $\mathcal{L}(s_0) = 1$. □

Next, we provide a short-sighted implementation of ESTIMATE-ϵ-distance(s), which, coupled with appropriate labeling functions, satisfies the conditions of Theorem 4.5. This implementation is outlined in Algorithm 2.

Algorithm 2 closely follows the labeling procedure of FLARES, with some major differences. First, as in Algorithm 1, all boolean label checks have been replaced by Bernoulli trials with probability $\mathcal{L}(s)$ (lines 7 an 19). Second, labeling is done by modifying the ϵ-distances of states, instead of assigning hard labels (lines 26 and 28). Third, the short-sighted horizon, h, is set to infinity with probability ψ, allowing $\mathcal{S}_{s,\pi}$ to be explored fully.

In more detail, the algorithm works as follows. Given a state s, Algorithm 2 expands all states in $\mathcal{S}_{s,\pi}$ up to distance $2h$, and checks if all of these are ϵ-consistent (lines 15-16); the notion of distance to use is specified by the function w (see line 20). If all of the states found are ϵ-consistent, the algorithm then modifies ϵ-distance estimates according to the distance—from s—at which the state was first found (lines 23-28). If it turns out that $\mathcal{S}_{s,\pi}$ lies completely within the horizon $2h$ (when variable *all* is true), then $\mathfrak{d}_\epsilon(s')$ is set to ∞ for all states found, since this condition is the usual requirement for correctly hard-labeling states.

Note that, in the case where only finite ϵ-distances can be assigned (line 28), our distance estimate slightly departs from Definition 4.3. A more accurate estimate would be $\mathfrak{d}_\epsilon(s') \leftarrow 2t - d$, considering that there are ϵ-consistent states at distances t to $2t$ from the initial state s. Similarly, we could also have assigned ϵ-distances for all states found up to distance $2t$, instead of only for those at distance $\leq t$ (line 27). However, we took the more conservative approach shown here because it is equivalent to a more

Algorithm 2: A depth limited procedure to compute ϵ-distances.

```
ESTIMATE-$\epsilon$-distance
   input : $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, C, s_0, \mathcal{G} \rangle$, $s$, $w$, $\epsilon$, $\psi$, $t$
1  no-high-res ← true
2  open ← EMPTY-STACK
3  closed ← EMPTY-STACK
4  all ← true
5  $z \sim Bernoulli(\psi)$
6  $h \leftarrow [z = 0] \cdot t + [z = 1] \cdot \infty$
7  if $[x \sim Bernoulli(\mathcal{L}(s)) = 1]$ then
8      open.PUSH($\langle s, 0 \rangle$)
9  while open ≠ EMPTY-STACK do
10     $\langle s, d \rangle \leftarrow$ open.POP()
11     if $d > 2h$ then
12         all ← false
13         continue
14     closed.PUSH($\langle s, d \rangle$)
15     if $s$.RESIDUAL() > $\epsilon$ then
16         no-high-res ← false
17     $a \leftarrow$ GREEDY-ACTION($s$)
18     for $s' \in \{s' \in \mathcal{S} | \mathcal{T}(s, a, s') > 0\}$ do
19         if $([x \sim Bernoulli(\mathcal{L}(s)) = 0]$
              $\vee\ h = \infty) \wedge s' \notin$ closed then
20             open.PUSH($(\langle s', d + w(\langle s, s' \rangle) \rangle)$)
21         else if $\mathfrak{d}_\epsilon(s') \neq \infty \wedge s' \notin$ closed then
22             all = false
23 if no-high-res then
24     for $\langle s', d \rangle \in$ closed do
25         if all then
26             $\mathfrak{d}_\epsilon(s') = \infty$
27         else if $d \leq t$ then
28             $\mathfrak{d}_\epsilon(s') = t - d$
29 else
30     while closed ≠ EMPTY-STACK do
31         $\langle s', d \rangle$ = closed.POP()
32         BELLMAN-UPDATE($s$)
```

robust version of FLARES; one that labels the same set of states, but that uses soft labels instead. This allows our method to explore, with some probability, states that have already been labeled. As it turns out, we experimented with the more accurate ϵ-distance estimates and its empirical performance was worse than FLARES'.

Finally, what are good labeling functions to use? Intuitively, we want increasing functions of the ϵ-distance, to encourage sampling towards states that are "more likely" to be far away from convergence. In our experiments we consider the following labeling functions, for the case when $t > 0$:

- **Linear**: $\mathcal{L}(s) \triangleq \frac{\beta - \alpha}{t} \mathfrak{d}_\epsilon(s) + \alpha$
- **Logistic**: $\mathcal{L}(s) \triangleq \frac{1}{1 + \frac{1-\alpha}{\alpha} \exp\{-\frac{1}{t} \ln \frac{(1-\alpha)\beta}{\alpha(1-\beta)} \cdot \mathfrak{d}_\epsilon(s)\}}$
- **Exponential**: $\mathcal{L}(s) \triangleq \alpha \exp\{\frac{1}{t} \ln \frac{\beta}{\alpha} \cdot \mathfrak{d}_\epsilon(s)\}$

where α and β are parameters that represent the desired labeling probability for $\mathfrak{d}_\epsilon(s) = 0$ and $\mathfrak{d}_\epsilon(s) = t$, respectively. In all cases,

Algorithm	Exp. cost	Time (seconds)
SOFT-FLARES-T-LOG(2)	93.89 ± 1.01	**3.53**
FLARES(2)	94.47 ± 0.99	4.65
BRTDP	94.86 ± 0.98	13.08
SSIPP(8)	95.26 ± 1.00	17.42
HDP(2)	95.64 ± 1.01	5.78
LRTDP	96.28 ± 1.04	7.57

Table 1: Expected cost and total planning of several planning algorithms on the sailing domain (middle-goal).

Algorithm	Exp. cost	Time (seconds)
BRTDP	180.74 ± 1.37	78.67
FLARES(2)	180.84 ± 1.39	12.58
HDP(1,0)	180.86 ± 1.39	31.11
LRTDP	180.96 ± 1.36	18.72
SOFT-FLARES-T-EXP(2)	181.04 ± 1.41	**9.99**
SSIPP(8)	181.39 ± 1.39	74.29

Table 2: Expected cost and total planning of several planning algorithms on an instance of the sailing domain (corner-goal).

Algorithm	Exp. cost	Time (seconds)
SOFT-FLARES-T-EXP(3)	27.27 ± 0.19	4.34
LRTDP	27.30 ± 0.19	12.27
HDP(3,0)	27.41 ± 0.20	**4.12**
BRTDP	27.52 ± 0.19	9.27
FLARES(3)	27.52 ± 0.20	6.49
SSIPP(8)	28.06 ± 0.20	30.46

Table 3: Expected cost and total planning of several planning algorithms on the racetrack domain (ring-5).

Algorithm	Exp. cost	Time (seconds)
SOFT-FLARES-T-LOG(3)	11.55 ± 0.05	**0.33**
FLARES(2)	11.55 ± 0.05	0.91
SSIPP(4)	11.58 ± 0.05	2.03
LRTDP	11.61 ± 0.05	58.62
BRTDP	11.64 ± 0.05	4.35
HDP(3,0)	11.65 ± 0.05	0.43

Table 4: Expected cost and total planning of several planning algorithms on the racetrack domain (square-4).

we assume that $\mathcal{L}(s) = 0$ if $\mathfrak{d}_\epsilon(s) < 0$ and $\mathcal{L}(s) = \beta$ if $\mathfrak{d}_\epsilon(s) \geq t$. Note that, under the provided definition of ESTIMATE-ϵ-DISTANCE, the FLARES label described in Eq. (10) can now be succinctly written as $\mathcal{L}(s') \triangleq [\mathfrak{d}_\epsilon(s') \geq t]$.

5 EXPERIMENTS

In this section we empirically evaluate the use of soft labeling for approximately solving SSPs, denoting the combination of Algorithms 1 and 2 as SOFT-FLARES. The goal of these experiments was to demonstrate that the general soft labeling framework can produce algorithms competitive with state-of-the-art methods, both in terms of expected cost and total planning time. We compared different variants of SOFT-FLARES to several SSP solvers: RFF [20], LRTDP [4], FLARES [16], HDP [3], BRTDP [15], and SSIPP [23]. We did not perform extensive experiments with LABELED-SSIPP, since preliminary experiments indicate that run time was never better than that of LRTDP. Similarly, we did not compare with VPI-RTDP [18] as it seems to be easily affected by the quality of initial upper bounds; in our experiments we have been unable to reproduce the results in [18].

We use the notation ALGORITHM(X) to denote the short-sighted horizon, X, used by the algorithm. In the case of SOFT-FLARES and SSIPP, the notation ALGORITHM-DIST-LABEL; refers to a distance function, DIST, (D for depth, T for trajectory probability, or P for plausibility), and a label function, LABEL, (LINear, LOGistic, or EXPonential). In all cases we used $\alpha = 0.1$ and $\beta = 0.9$ for SOFT-FLARES. For SOFT-FLARES, we also set $\psi = 0$, since we will evaluate the quality of the resulting policies empirically. We used the h_{min} heuristic [4], pre-computed for all states before planning started. We evaluated different parameterizations of the algorithms with distances from 0 to 4 for HDP, FLARES and SOFT-FLARES, distances in $\{1, 2, 4, 8\}$ for SSIPP, and values of $\rho = 2^{-5}$ and $\rho = 2^{-4}$ for TRAJECTORY-BASED-SSIPP [22]. For each algorithm, we report the results of the parameterization resulting in the lowest planning time, under the constraint that the obtained expected cost is within two standard errors of the best observed one (if possible). In Section 6, we discuss the sensitivity of SOFT-FLARES to its parameters values in more detail.

All experiments were performed on Xeon E5-2680 v4 @ 2.40GHz computers. The performance of a planner is evaluated by running simulations of the partial policy implied by the algorithm's action selection, and computing the resulting expected cost and total time spent on planning. We reset any internal state of the algorithms before each simulation starts, to evaluate their performance in a one-shot planning task. Note that this is harder than typical competition settings, where planners are allowed to reuse computation from previous simulations. Actions are selected greedily on the current value estimates, and we allow the algorithm to re-plan if necessary, adding the accrued time to the total. For SSIPP, the algorithm re-plans before each action, for HDP re-planning is done as described by [3], and for SOFT-FLARES it is done whenever a soft-label check fails.[2]

5.1 Sailing Domain

Our first evaluation benchmark is the sailing problem [11]. We evaluated on two instances of this domain, both with size 40×40 (12,801 states), differing in the goal location (corner or middle of the grid). The wind transition probability is such that the direction stays the same with probability 0.3, changes by one unit (clockwise or counterclockwise) with probability 0.2, and by two units with probability 0.15. We consider the performance of the algorithms when there is no time limit per action, considering the following

[2] Code to reproduce these experiments available at https://github.com/luisenp/mdp-lib.

Algorithm	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10
TRIANGLE-TIREWORLD										
SOFT-FLARES	50	50	50	50	50	50	50	49	49	41
RFF	50	50	50	50	50	50	50	0	0	0
TRAJECTORY-BASED-SSIPP	50	50	48	46	46	38	40	33	42	31
EX-BLOCKSWORLD										
SOFT-FLARES	45	15	17	21	50	48	50	27	23	1
RFF	31	7	25	10	50	12	41	6	5	0
TRAJECTORY-BASED-SSIPP	0	0	31	26	50	46	0	0	0	0

Table 5: Number of runs in which planners were able to successfully reach the goal for two IPPC'08 domains.

stopping criteria (whichever happens first): successful label checks for FLARES, SOFT-FLARES, and HDP; 1000 trials for LRTDP, BRTDP, FLARES, and SOFT-FLARES; a single simulated trial reaching a goal for SSIPP. Tables 1 and 2 show the expected costs and total planning times of these experiments, averaged over 1000 simulations; standard errors are also shown for the expected cost (the variance for the planning time was negligible). Note that all planners are within statistical error of the optimal performance (using the expected cost of LRTDP as reference), so meaningful distinctions can only be established in terms of planning time. In both of the problem instances considered, a parameterization of SOFT-FLARES was considerably faster than all other planners. Additionally, in both cases the best parameterization of SOFT-FLARES involved the trajectory probability distance, although other parameterizations achieved similar performance.

5.2 Racetrack Domain

Our second evaluation benchmark is the racetrack domain, first proposed in [1]. We modify the domain so that, in addition to the probability of slipping, all actions have some probability that the acceleration changes from the intended one by 1 unit, in any of the directions chosen uniformly at random; for example, accelerating north can also result in accelerating north-west, north-east, or in no acceleration. Note that similar modifications have been introduced in the past to increase the complexity of the problem [15, 16]. We used a probability of 0.20 for slipping, and a probability of 0.10 for randomly changing accelerations. We experimented with two problems instances, one with 92,909 states (ring-5) and one with 400,270 states (square-4). The results are shown in Tables 3 and 4, respectively, which show the expected costs and total planning times of these experiments, averaged over 1000 simulations. In the two problem instances considered, a parameterization of SOFT-FLARES was among the best two planners in terms of total planning time, being outperformed only by a slight margin by the HDP algorithm. As in the case of the sailing domain, the best parameterization involved the trajectory probability distance function.

5.3 International Planning Competition Domains

We assessed the scalability of SOFT-FLARES to problems with very large state spaces, using two domains from the International Planning Competition held in 2008 (the last competition involving goal-based MDPs) [7]. We used problems 1-10 from two of the domains: TRIANGLE-TIREWORLD and EX-BLOCKSWORLD; in contrast with the domains explored in the previous sections, the state space of the larger instances of these domains consist of billions of states. As typical in competition settings, we gave planners 20 minutes to successfully complete 50 runs of each problem. For SOFT-FLARES, we used $t = 2$, an exponential labeling function and the inadmissible FF heuristic [5]. We compare the performance of SOFT-FLARES with our implementation of RFF [20], the winner of IPPC'08, and with trajectory-based SSIPP [22], using $\rho = 0.25$ and the h_{add} heuristic (SSIPP code provided by the original author); all experiments were run on the same machine. The results, shown in Table 5, demonstrate that SOFT-FLARES is able to scale to very large problems, outperforming state-of-the-art planners in two probabilistically interesting domains [13]. Crucially, it outperforms RFF, *without relying on a classical planner to speed up computation*, providing convincing evidence that soft labeling is a promising framework for scalable and performant probabilistic planning. Note that for TRAJECTORY-BASED-SSIPP, the original work reports 50 in every triangle-tireworld instance [22], but we could not reproduce these results using the original code.

6 DISCUSSION AND POTENTIAL DRAWBACKS

The results described in the previous section offer evidence that our soft labeling framework can be used to create planners with near-optimal performance and competitive planning times. However, our work opens ups some research questions for which we do not have a definite answer yet, and, thus, it would be useful for the reader to be aware of some of the potential aspects that can be improved in our proposed approach.

One issue is the question of how to select the parameters for the algorithm, namely, the horizon t, and the labeling and distance functions. Tables 6 and 7 show the results obtained with different parameterizations of SOFT-FLARES in one instance of the sailing domain, and one of the racetrack, respectively; the parameterizations reported in Section 5 are highlighted in bold font. The choice of parameters can have a significant impact on the quality of the final results, both in terms of planning time and expected cost. In particular, the choice of distance function can increase the running time up to an order of magnitude, without significant improvement in policy quality. Out of the distance functions considered, the depth-based distance ($w(\langle s, s' \rangle = 1)$) resulted in much slower

Parameterization	Exp. cost	Time (seconds)
PLAUS-LIN(4)	178.51 ± 1.37	73.91
DEPTH-EXP(2)	179.37 ± 1.37	15.81
TRAJ-LOG(2)	181.04 ± 1.41	9.99
TRAJ-EXP(2)	181.04 ± 1.41	9.99
TRAJ-LIN(2)	181.04 ± 1.41	10.48
DEPTH-EXP(4)	182.47 ± 1.40	34.54
PLAUS-LOG(4)	185.03 ± 1.47	72.85

Table 6: Expected cost and total planning time of several parameterizations of SOFT-FLARES on the sailing domain (corner-goal).

Parameterization	Exp. cost	Time (seconds)
PLAUS-EXP(4)	27.25 ± 0.18	7.40
TRAJ-EXP(3)	27.27 ± 0.19	4.34
DEPTH-LIN(2)	27.37 ± 0.19	7.84
DEPTH-EXP(4)	27.42 ± 0.19	18.02
TRAJ-LIN(3)	27.58 ± 0.20	5.30
TRAJ-LOG(3)	27.60 ± 0.19	4.70
DEPTH-LOG(4)	27.78 ± 0.20	18.54

Table 7: Expected cost and total planning time of several parameterizations of SOFT-FLARES on the racetrack domain (ring-5).

performance on both of these problems, even with smaller horizons. For an example, see DEPTH-EXP(2) in Table 6 and DEPTH-LIN(2) in Table 7. The plausibility distance function had poor running time on the sailing domain, but better performance on the racetrack. In general, the results suggest that the trajectory probability distance can be a good choice, but this may be problem dependent (e.g, if transition probabilities are markedly non uniform, plausibilities could be better).

The choice of labeling function also has influence on the performance of the algorithm, as evidenced by the differences between TRAJ-LIN(2) and TRAJ-EXP(2) in Table 6, PLAUS-LIN(4) and PLAUS-LOG(4) in Table 6, and TRAJ-EXP(3) and TRAJ-LIN(3) in Table 7. Note that in one case (PLAUS-LIN(4) vs. PLAUS-LOG(4) in Table 6), the choice made a statistical significance difference in expected cost (p-value of 0.0001). On the other hand, when using the trajectory based distance, we did not find significant differences between the exponential and logistic labeling functions (the linear labeling function was found to be slightly slower). This suggest that the combination of trajectory probability distance with exponential/logistic labeling function can be effective in practice, but, given the heuristic nature of these functions, we caution that the results are likely to be problem dependent.

The other aspect of our approach that requires further investigation is the trade-off between theoretical and empirical results. The results of Theorem 4.4 imply that, unlike their deterministic counterparts, soft labeling allows short-sighted planners to naturally leverage any additional planning time to increase policy quality. To empirically confirm this property, we experimented with a soft analogue of FLARES, which can be obtained under SOFT-FLARES using a depth based distance and the label function $\mathcal{L}(s') \triangleq \beta \cdot [\mathfrak{d}_\epsilon(s') \geq t]$. We used a similar setting to Section 5, allowing the planner to run until the initial state is sampled as labeled. While this means the algorithm is not necessarily running until optimality, the probabilistic labels should allow the planner to run for longer than regular FLARES, and result in lower expected costs. We experimented with values of $\beta \in [0.1, 0.2, ..., 0.9]$, and performed 4000 simulations, focusing on the case of horizon $t = 1$. In general, we observed that the probabilistic labels generally resulted in lower expected costs than those obtained with FLARES, although only statistically significant in a few cases. In particular, in the sailing domain (size 40, middle goal), all values of $\beta \in \{0.1, 0.3, 0.5, 0.7\}$ improve the expected cost by more than 2%, significant at a 0.05 level (p-value < 0.005). Likewise, in racetrack instance ring-5, a value $\beta = 0.3$ improves the expected cost by 1.2% (p-value of 0.018). We emphasize that these improvements are solely due to the use of probabilistic labels. However, as expected, they come with an increase in planning time, which in these experiments ranges from 15% to 62%.

On the other hand, we have had less success translating the results of Theorem 4.5 into empirical improvements. Ideally, we would expect that an optimal instantiation of SOFT-FLARES, using $\psi \neq 0$, would be faster than LRTDP. Unfortunately, we have not seen any significant difference in running time between LRTDP and the optimal version of SOFT-FLARES, even with different choices of ψ. It is possible that a more refined implementation of ESTIMATE-ϵ-DISTANCE and other labeling functions would result in a more efficient optimal solver, but we leave this for future work.

7 CONCLUSION

We introduced *soft labeling*, a planning framework that generalizes state labeling in SSPs. Crucially, soft labeling exploits the computational advantages of short-sightedness, while still allowing the possibility of complete exploration of the state space, and maintaining theoretical guarantees. Our definition of soft labeling exploits the concept of ϵ-distance, a proposed heuristic measure for quantifying how close the value of a state is to ϵ-consistency. We introduced an efficient algorithm to estimate ϵ-distances, and combined it with a soft-labeled variant of RTDP to produce an instance of our framework, the SOFT-FLARES algorithm. We compare SOFT-FLARES to several popular short-sighted solvers in four well-known benchmarks, showing that it can produce policies with similar or better quality, and shorter planning times. We also discuss some of the potential drawbacks of our approach. One possible avenue to address these, is the development of distance and labeling functions that are connected in a principled manner to the value of the state to be labeled. With the goal of illustrating the benefits of our framework, in this work we have provided a variety of heuristics that can be effective in practice, yet their performance can be problem dependent. We envision that the most benefit of this flexible framework is yet to be obtained, and developing more refined labeling and distance functions is a promising direction of future work.

ACKNOWLEDGEMENT

Support for this work was provided in part by the National Science Foundation under grant IIS-1524797.

REFERENCES

[1] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. 1995. Learning to Act Using Real-Time Dynamic Programming. *Artificial Intelligence* 72 (1995), 81–138.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. 1991. An analysis of stochastic shortest path problems. *Mathematics of Operations Research* 16, 3 (1991), 580–595.

[3] Blai Bonet and Héctor Geffner. 2003. Faster Heuristic Search Algorithms for Planning with Uncertainty and Full Feedback. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. Acapulco, Mexico, 1233–1238.

[4] Blai Bonet and Héctor Geffner. 2003. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling*. Trento, Italy, 12–21.

[5] Blai Bonet and Héctor Geffner. 2005. mGPT: A probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research* 24 (2005), 933–944.

[6] Blai Bonet and Héctor Geffner. 2006. Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In *Proceedings of the Sixteenth International Conference on International Conference on Automated Planning and Scheduling*. 142–151.

[7] Daniel Bryce and Olivier Buffet. 2008. Sixth international planning competition: Uncertainty part. In *Proceedings of the Sixth International Planning Competition*.

[8] Mohammad Hajian, Emanuel Melachrinoudis, and Peter Kubat. 2016. Modeling wildfire propagation with the stochastic shortest path: A fast simulation approach. *Environmental Modelling & Software* 82 (2016), 73–88.

[9] Qilong Huang, Qing-Shan Jia, and Xiaohong Guan. 2018. Robust scheduling of EV charging load with uncertain wind power integration. *IEEE Transactions on Smart Grid* 9, 2 (2018), 1043–1054.

[10] Thomas Keller and Malte Helmert. 2013. Trial-based Heuristic tree search for finite horizon MDPs. In *Proceedings of the Twenty-Third International Conference on International Conference on Automated Planning and Scheduling*. 135–143.

[11] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo planning. In *Proceedings of the Seventeenth European conference on Machine Learning*. 282–293.

[12] Andrey Kolobov, Peng Dai, Mausam Mausam, and Daniel S. Weld. 2012. Reverse iterative deepening for finite-horizon MDPs with large branching factors. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling*.

[13] Iain Little and Sylvie Thiebaux. 2007. Probabilistic planning vs. replanning. In *Proceedings of the ICAPS'07 Workshop on the International Planning Competition: Past, Present and Future*.

[14] Michael L. Littman. 1997. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. Providence, Rhode Island, 748–754.

[15] H. Brendan McMahan, Maxim Likhachev, and Geoffrey J. Gordon. 2005. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proceedings of the 22nd international conference on Machine learning*. 569–576.

[16] Luis Pineda, Kyle Hollins Wray, and Shlomo Zilberstein. 2017. Fast SSP Solvers Using Short-Sighted Labeling. In *Proceedings of the Thirty-First Conference on Artificial Intelligence*. 3629–3635.

[17] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA.

[18] Scott Sanner, Robby Goetschalckx, Kurt Driessens, and Guy Shani. 2009. Bayesian Real-time Dynamic Programming. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*.

[19] Trey Smith and Reid Simmons. 2006. Focused real-time dynamic programming for MDPs: squeezing more out of a heuristic. In *Proceedings of the 21st National Conference on Artificial intelligence*, Vol. 2. 1227–1232.

[20] Florent Teichteil-Königsbuch, Ugur Kuter, and Guillaume Infantes. 2010. Incremental plan aggregation for generating policies in MDPs. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*. Toronto, Canada, 1231–1238.

[21] Felipe W. Trevizan and Manuela M. Veloso. 2012. Short-sighted stochastic shortest path problems. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling*. Atibaia, Brazil, 288–296.

[22] Felipe W. Trevizan and Manuela M. Veloso. 2012. Trajectory-Based Short-Sighted Probabilistic Planning. In *Proceedings of Neural Information Processing Systems*. Lake Tahoe, Nevada, 3257–3265.

[23] Felipe W. Trevizan and Manuela M. Veloso. 2014. Depth-based short-sighted stochastic shortest path problems. *Artificial Intelligence* 216 (2014), 179–205.

[24] Kyle Hollins Wray, Luis Pineda, and Shlomo Zilberstein. 2016. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 517–523.