Revisiting Risk-Sensitive MDPs: New Algorithms and Results*

Ping Hou
Department of Computer Science
New Mexico State University
phou@cs.nmsu.edu

William Yeoh
Department of Computer Science
New Mexico State University
wyeoh@cs.nmsu.edu

Pradeep Varakantham
School of Information Systems
Singapore Management University
pradeepv@smu.edu.sg

Abstract

While *Markov Decision Processes* (MDPs) have been shown to be effective models for planning under uncertainty, the objective to minimize the expected cumulative cost is inappropriate for high-stake planning problems. As such, Yu, Lin, and Yan (1998) introduced the *Risk-Sensitive MDP* (RS-MDP) model, where the objective is to find a policy that maximizes the probability that the cumulative cost is within some user-defined cost threshold. In this paper, we revisit this problem and introduce new algorithms that are based on classical techniques, such as depth-first search and dynamic programming, and a recently introduced technique called *Topological Value Iteration* (TVI). We demonstrate the applicability of our approach on randomly generated MDPs as well as domains from the ICAPS 2011 *International Probabilistic Planning Competition* (IPPC).

Introduction

Markov Decision Processes (MDPs) have been shown to be effective models for planning under uncertainty. Typically, MDP solvers find policies that minimize the expected cumulative cost (or, equivalently, maximize the expected cumulative reward). While such a policy is good in the expected case, there is a small chance that it might result in an exorbitantly high cost. Therefore, it is not suitable in high-stake planning problems where exorbitantly high costs *must* be avoided. For example, imagine a time-sensitive logistics problem, where a delivery truck needs to plan its route, whose travel times are stochastic, in order to reach its destination before a strict deadline (Ermon et al., 2012). Aside from this logistics example, other high-stake planning situations include environmental crisis situations (Cohen et al., 1989; Blythe, 1999), business decision situations (Murthy et al., 1999; Goodwin, Akkiraju, and Wu, 2002), planning situations in space (Pell et al., 1998; Zilberstein et al., 2002), and sustainable planning situations (Ermon et al., 2011).

With this motivation in mind, Yu, Lin, and Yan (1998) introduced the *Risk-Sensitive MDP* (RS-MDP) model, where the objective is to find a policy π that maximizes the probability $Pr(c^{\mathcal{T}(\pi)}(s_0) \leq \theta_0)$, where $c^{\mathcal{T}(\pi)}(s_0)$ is the cumulative cost or travel time of the policy and θ_0 is the cost threshold or deadline. They also introduced a *Value Iteration* (VI) like algorithm to solve the problem. Unfortunately, their algorithm, like VI, cannot scale to large problems as it needs to perform Bellman updates for all states in each iteration.

As such, in this paper, we revisit RS-MDPs and develop new RS-MDP algorithms that are faster than VI. Specifically, we introduce algorithms that are based on classical techniques, such as depth-first search and dynamic programming, and a recently introduced technique called Topological Value Iteration (Dai et al., 2011). We demonstrate the applicability of our approach on randomly generated MDPs as well as domains from the ICAPS 2011 *International Probabilistic Planning Competition* (IPPC).

MDP Model

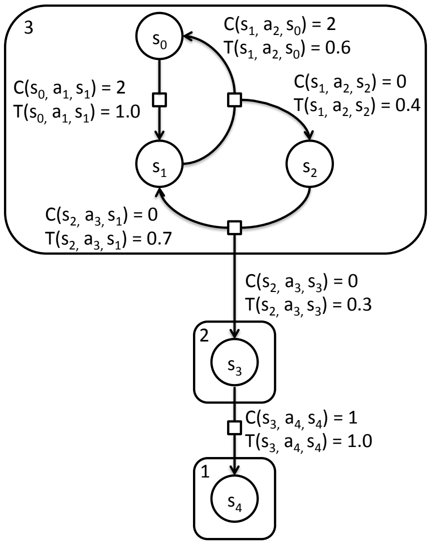
A *Goal-Directed MDP* (GD-MDP) is represented as a tuple $\mathbf{P} = \langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G} \rangle$. It consists of a set of states $\mathbf{S}$; a start state $s_0 \in \mathbf{S}$; a set of actions $\mathbf{A}$; a transition function $\mathbf{T} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$ that gives the probability $T(s, a, \hat{s})$ of transitioning from state s to $\hat{s}$ when action a is executed; a cost function $\mathbf{C} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, \infty)$[1] that gives the cost $C(s, a, \hat{s})$ of executing action a in state s and arriving in state $\hat{s}$; and a set of goal states $\mathbf{G} \subseteq \mathbf{S}$, which are terminal, that is, $T(s_g, a, s_g) = 1$ and $C(s_g, a, s_g) = 0$ for all goal states $s_g \in \mathbf{G}$ and actions $a \in \mathbf{A}$. In this paper, we will focus on GD-MDPs and will thus use the term MDPs to refer to GD-MDPs. Notice that our definition of GD-MDPs allows problems that do not lie in the *Stochastic Shortest-Path MDP* (SSP-MDP) class (Bertsekas, 2000) because our definition does not require the existence of a proper policy. A proper policy is a policy with which an agent can reach a goal state from any state with probability 1.

Strongly Connected Components (SCCs)

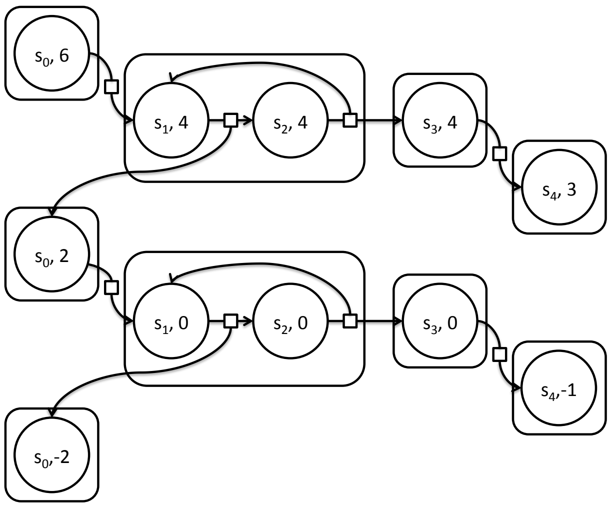
The state space of an MDP can be visualized as a directed hyper-graph called a *connectivity graph*. Figure 1(a) shows the connectivity graph of our example MDP, where nodes

*A part of this research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA). We also thank Yaxin Liu for sharing with us his implementation of FVI.

[1]With a slight modification, our algorithms can also be applied on GD-MDPs with negative cost as long as the cost function does not form negative cycles in the connectivity graph.



(a) Example MDP



(b) Example Augmented MDP

Figure 1: Connectivity Graphs

correspond to states (denoted by circles) and hyper-edges correspond to actions (denoted by squares) and transitions (denoted by arrows).[2] We will use this example as our running example throughout this paper. The subgraph rooted at a start state is called a *transition graph*.

It is possible to partition the connectivity or transition graphs into *Strongly Connected Components* (SCCs) in such a way that they form a *Directed Acyclic Graph* (DAG) (Tarjan, 1972; Bonet and Geffner, 2003; Dai et al., 2011). More formally, an SCC of a directed graph $G = (V, E)$ is a maximal set of vertices $Y \subseteq V$ such that every pair of vertices u and v in Y are reachable from each other. Since the SCCs form a DAG, it is impossible to transition from a state in a downstream SCC to a state in an upstream SCC. Each SCC is denoted by a rounded rectangle in Figure 1. We call this DAG an *SCC transition tree*.

MDP Algorithms

An *MDP policy* $\pi : \mathbf{S} \to \mathbf{A}$ is a mapping from states to actions. A common objective is to find a policy π with the minimum expected cost $\mathcal{C}^\pi(s_0)$, where

$$\mathcal{C}^\pi(s) = \sum_{\hat{s} \in \mathbf{S}} T(s, \pi(s), \hat{s}) \big[C(s, \pi(s), \hat{s}) + \mathcal{C}^\pi(\hat{s}) \big] \qquad (1)$$

for all states $s \in \mathbf{S}$.

Value Iteration (VI)

Value Iteration (VI) (Bellman, 1957) is one of the fundamental approaches to find an optimal expected cost policy. It uses a function $\mathcal{C}$ to represent expected costs. The expected cost of the policy is the expected cost $\mathcal{C}(s_0)$ for start state s_0, which is calculated using the Bellman equation:

$$\mathcal{C}(s) = \min_{a \in \mathbf{A}} \sum_{\hat{s} \in \mathbf{S}} T(s, a, \hat{s}) \big[C(s, a, \hat{s}) + \mathcal{C}(\hat{s}) \big] \qquad (2)$$

The action chosen for the policy for each state s is then the one that minimizes $\mathcal{C}(s)$. A single update using this equation is called a *Bellman update*. In each iteration, VI performs a Bellman update on each state. The difference between the expected cost of a state in two consecutive iterations is called the residual of that state and the largest residual is called the *residual error*. The algorithm terminates when the values converge, that is, the residual error is less than a user-defined threshold ϵ. Lastly, VI can be optimized by only considering the set of states reachable from the start state.

Topological VI (TVI)

VI suffers from a limitation that it updates each state in every iteration even if the expected cost of states that it can transition to remain unchanged. *Topological VI* (TVI) (Dai et al., 2011) addresses this limitation by repeatedly updating the states in only *one* SCC until their values converge before updating the states in another SCC. Since the SCCs form a DAG, states in an SCC only affect the states in upstream SCCs. Thus, by choosing the SCCs in reverse topological sort order, it no longer needs to consider SCCs whose states have converged in a previous iteration. Figure 1(a) shows the indices of the SCCs in reverse topological sort order on the upper left corner of the rounded rectangles. Like VI, TVI can also be optimized by only considering the set of states reachable from the start state.

Risk-Sensitive MDP Model

A *Risk-Sensitive MDP* (RS-MDP) is defined by the tuple $\langle \mathbf{P}, \mathbf{\Theta}, \theta_0 \rangle$, where $\mathbf{P}$ is an MDP, $\mathbf{\Theta}$ is a set of possible cost

[2]We call them hyper-edges as a state can transition to *multiple* successor states.

thresholds, and $\theta_0 \in \mathbf{\Theta}$ is the user-defined cost threshold. The objective is to find a policy π that maximizes the probability that the cumulative cost $c^{\mathcal{T}(\pi)}(s_0)$ is no greater than the cost threshold θ_0 (Yu, Lin, and Yan, 1998):

$$\underset{\pi}{\operatorname{argmax}}\ Pr(c^{\mathcal{T}(\pi)}(s_0) \leq \theta_0) \tag{3}$$

The cumulative cost $c^{\mathcal{T}(\pi)}(s_0)$ of a trajectory $\mathcal{T}(\pi) = \langle s_0, s_1 = \pi(s_0), s_2 = \pi(s_1), \ldots \rangle$, formed by executing an MDP policy π, is defined as follows:

$$c^{\mathcal{T}(\pi)}(s_0) = c^{\mathcal{T}(\pi)}(s_0, \infty) \tag{4}$$

$$c^{\mathcal{T}(\pi)}(s_0, H) = \sum_{t=0}^{H} C(s_t, s_{t+1}) \tag{5}$$

where s_t is the state in the t-th time step.

The optimal policy for an MDP often does not depend on the time step or the accumulated cost thus far. In contrast, an optimal policy for an RS-MDP does depend on the accumulated cost thus far. For example, one can take riskier actions when the accumulated cost is small, but should avoid them when the accumulated cost is close to the threshold. Therefore, instead of a mapping of states to actions, an *RS-MDP policy* $\pi : \mathbf{S} \times \mathbf{\Theta} \rightarrow \mathbf{A}$ is a mapping of augmented states $(s, \theta \mid s \in \mathbf{S}, \theta \in \mathbf{\Theta})$ to actions $a \in \mathbf{A}$. The threshold $\theta = \theta_0 - c^{\mathcal{T}(\pi)}(s_0, t)$ is the amount of unused cost, where $c^{\mathcal{T}(\pi)}(s_0, t)$ is the accumulated cost thus far up to the current time step t. In this paper, we focus on finding *deterministic* policies, that is, policies that always return the same action for an augmented state. We show in the theoretical results section that deterministic policies are optimal.

We use the notation $P^{\pi}(s, \theta)$ to denote the *reachable probability*, that is, the probability $Pr(c^{\mathcal{T}(\pi)}(s) \leq \theta)$ that the accumulated cost of starting from state s is no larger than a cost threshold θ with policy π. Thus, in solving an RS-MDP, the goal is to find a policy π^* such that:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, P^{\pi}(s_0, \theta_0) \tag{6}$$

We refer to the maximum probability value corresponding to π^* as $P(s_0, \theta_0)$.

Solution Approach

Similar to the Bellman equation to back up value functions in regular MDPs (Bellman, 1957), one can characterize the relationship between the different augmented states of an RS-MDP with the following system of equations:

$$P(s, \theta) = \max_{a \in \mathbf{A}} \sum_{\hat{s} \in \mathbf{S}} P(s, a, \hat{s}, \theta) \tag{7}$$

$$P(s, a, \hat{s}, \theta) =$$

$$\begin{cases} 0 & \text{if } \theta < C(s, a, \hat{s}) \\ T(s, a, \hat{s}) & \text{if } \hat{s} \in \mathbf{G},\, \theta \geq C(s, a, \hat{s}) \\ T(s, a, \hat{s}) \cdot P(\hat{s}, \theta - C(s, a, \hat{s})) & \text{if } \hat{s} \notin \mathbf{G},\, \theta \geq C(s, a, \hat{s}) \end{cases}$$

where $P(s, a, \hat{s}, \theta)$ is the reachable probability from augmented state (s, θ) assuming that one takes action a from state s and transitions to successor state $\hat{s}$. For each state-action-successor-threshold tuple $(s, a, \hat{s}, \theta)$, there are the following three cases:

Algorithm 1: TVI-DFS(θ_0)

```
1 $Y$ = FIND-SCCS($s_0, \theta_0$)
2 for SCCs $y_i \in Y$ with indices $i = 1$ to $n$ do
3     UPDATE-SCC($y_i$)
```

Procedure Update-SCC(y_i)

```
4 for $(s, \theta) \in y_i$ do
5     $P(s, \theta) = 0$
6 repeat
7     $residual = 0$
8     for $(s, \theta) \in y_i$ do
9         $P'(s, \theta) = P(s, \theta)$
10        UPDATE($s, \theta$)
11        if $residual < |P(s, \theta) - P'(s, \theta)|$ then
12            $residual = |P(s, \theta) - P'(s, \theta)|$
13 until $residual < \epsilon$;
```

- If the cost threshold θ is smaller than the action cost $C(s, a, \hat{s})$, then the successor can only be reached by exceeding the cost threshold. Thus, the reachable probability is 0.
- If the successor is a goal state and the cost threshold is larger than or equal to the action cost $C(s, a, \hat{s})$, then the successor can be reached without exceeding the cost threshold. Thus, the reachable probability is the transition probability $T(s, a, \hat{s})$.
- If the successor is not a goal state and the cost threshold is larger than or equal to the action cost $C(s, a, \hat{s})$, then the successor can be reached without exceeding the cost threshold. Thus, the reachable probability can be recursively determined as the transition probability $T(s, a, \hat{s})$ multiplied by the reachable probability of a new augmented state $P(\hat{s}, \theta - C(s, a, \hat{s}))$.

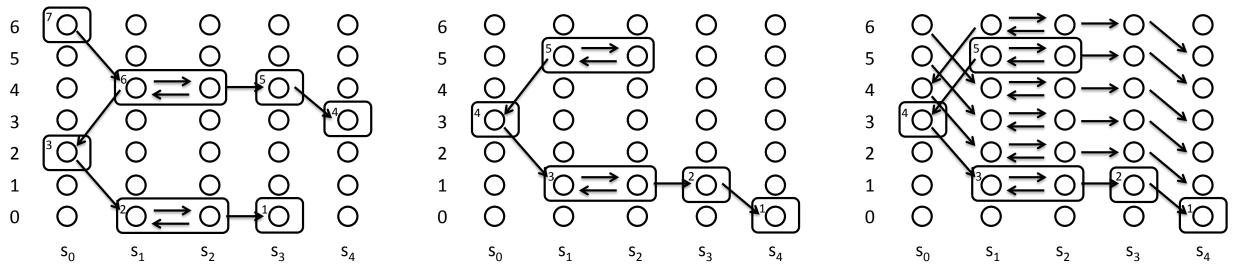
One can extract the optimal policy by taking the action that is returned by the maximization operator in Equation 7 for each augmented state (s, θ).

Augmented MDP

One can transform the MDP connectivity graph, where nodes correspond to states and hyper-edges correspond to actions and transitions, to an *augmented MDP* connectivity graph, where nodes now correspond to augmented states. This augmented MDP is a MAXPROB MDP (Kolobov et al., 2011), where the reward function is 1 for transitions that transition into a goal state and 0 otherwise. For our example MDP, Figure 1(b) shows the connectivity graph of the corresponding augmented MDP with s_0 as the start state, s_4 as the goal state, and an initial cost threshold $\theta_0 = 6$. Note that states that are in the same SCC in an MDP might not be in the same SCC in an augmented MDP. For example, states s_0, s_1, and s_2 are in the same SCC, but augmented states $(s_0, 6)$, $(s_1, 4)$, and $(s_2, 4)$ are not all in the same SCC.

TVI-DFS

We now introduce TVI-DFS—an algorithm based on TVI and *Depth-First Search* (DFS). At a high level, TVI-DFS is identical to TVI, except that it operates on an augmented



(a) TVI-DFS (b) TVI-DP (SCCs only) (c) TVI-DP

Figure 2: Transitions in the Augmented State Space

```
Procedure Update($s, \theta$)
14 $P^* = 0$
15 for $a \in \mathbf{A}$ do
16     $P_a = 0$
17     for $\hat{s} \in \mathbf{S} \mid T(s, a, \hat{s}) > 0$ do
18         if $\theta \geq C(s, a, \hat{s})$ then
19             if $\hat{s} \in \mathbf{G}$ then
20                 $P_a = P_a + T(s, a, \hat{s})$
21             else
22                 $\hat{\theta} = \theta - C(s, a, \hat{s})$
23                 $P_a = P_a + T(s, a, \hat{s}) \cdot P(\hat{s}, \hat{\theta})$
24     if $P_a > P^*$ then
25         $P^* = P_a$
26         $a^* = a$
27 $P(s, t) = P^*$
28 $\pi(s, t) = a^*$
```

MDP instead of a regular MDP and it uses Equation 7 to update the probabilities of each augmented state instead of using the Bellman equation to update the value of each state.

Algorithm 1 shows the pseudocode of the algorithm. TVI-DFS first partitions the augmented MDP state space into SCCs with Tarjan's algorithm (Tarjan, 1972), which traverses the connectivity graph in a depth-first manner and marks the SCC membership of each state (line 1). Tarjan's algorithm returns an SCC transition tree Y, where the SCC indices are in reverse topological sort order. It then performs a depth-first search, but instead of calling a recursive function in depth-first order, the algorithm updates the augmented states in the SCCs in reverse topological sort order (lines 2-3). This process is similar to popping elements of a stack that are pushed in depth-first order. For each SCC, the algorithm performs a VI-like update using Equation 7 until the residual of all augmented states, defined as the difference in the probability between subsequent iterations, are within ϵ (lines 6-13).

Figure 2(a) shows the resulting augmented MDP when one runs the TVI-DFS algorithm on our example MDP with s_0 as the start state, s_4 as the goal state, and an initial cost threshold $\theta_0 = 6$. Each circle represents an augmented state (s, θ), where the state s is shown on the x-axis and the threshold θ is shown on the y-axis. Each rounded rectangle represents an SCC and each arrow represents a possible transition after taking the sole action from that augmented state. For example, after taking an action in augmented state $(s_1, 4)$, one can transition to augmented states $(s_2, 4)$ or $(s_0, 2)$. Nodes without arrows represent unreachable augmented states and are thus not included in the SCC transition tree. The indices of the SCCs in reverse topological sort order are shown on the upper left corner of the rectangles. This augmented MDP is the same as the one shown in Figure 1(b) except that augmented states with thresholds $\theta < 0$ are omitted. Table 1(a) shows the probability $P(s, \theta)$ of each reachable augmented state (s, θ).

(a) TVI-DFS

θ		s_0	s_1	s_2	s_3	s_4
	6	0.167	-	-	-	-
	5	-	-	-	-	-
	4	-	0.167	0.417	0.300	-
θ	3	-	-	-	-	1.000
	2	0.000	-	-	-	-
	1	-	-	-	-	-
	0	-	0.000	0.000	0.000	-

(b) TVI-DP

θ		s_0	s_1	s_2	s_3	s_4
	6	0.167	0.306	0.514	0.300	1.000
	5	0.167	0.306	0.514	0.300	1.000
	4	0.167	0.167	0.417	0.300	1.000
θ	3	0.167	0.167	0.417	0.300	1.000
	2	-	0.167	0.417	0.300	1.000
	1	-	0.167	0.417	0.300	1.000
	0	-	-	-	-	1.000

Table 1: Augmented State Probabilities

One can slightly optimize this algorithm by integrating Tarjan's algorithm to find SCCs (line 1) with the procedure to update augmented states in the SCCs (line 3). Specifically,

- When Tarjan's algorithm finds a leaf SCC,[3] then TVI-DFS can pause Tarjan's algorithm, update the augmented states in that SCC, and then resume Tarjan's algorithm. For example, when Tarjan's algorithm finds SCC 1 (which contains $(s_3, 0)$) in Figure 2(a), TVI-DFS calls UPDATE-SCC on that SCC before backtracking to

[3]A leaf SCC is an SCC without any downstream SCCs.

Algorithm 2: TVI-DP(θ_0)

```
29 for $s_g \in \mathbf{G}$ do
30     $P(s_g, 0) = 1$
31 $Y$ = FIND-TRANSPOSED-SCCS($\mathbf{G}, 0$)
32 for $\theta = 0$ to $\theta_0$ do
33     for SCCs $y_i^\theta \in Y$ with indices $i = 1$ to $n$ do
34         UPDATE-SCC($y_i^\theta$)
35     for $s \in \mathbf{S} \mid (s, \theta) \notin Y$ do
36         $P(s, \theta) = P(s, \theta - 1)$
37         $\pi(s, \theta) = \pi(s, \theta - 1)$
```

SCC 2 (which contains $(s_1, 0)$ and $(s_2, 0)$).

- When Tarjan's algorithm backtracks to an SCC, all augmented states in downstream SCCs would have been updated. Thus, TVI-DFS can also update the augmented states in the SCC that it just backtracked to. For example, when Tarjan's algorithm backtracks to SCC 2, TVI-DFS calls UPDATE-SCC on that SCC since the augmented state in SCC 1 has already been updated.

This optimized version requires less memory as it does not need to represent the SCCs explicitly. We implement this optimized version in the experiments but provided the simpler pseudocode for the sake of clarity.

Additionally, one can optimize this algorithm further if the cost function $\mathbf{C} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow (0, \infty)$ does not return zero costs. In such a situation, each SCC contains exactly one augmented state (see Theorem 1). Thus, there is no need to use Tarjan's algorithm to find the SCCs and there is no need to check for convergence in each SCC.

TVI-DP

While TVI-DFS is efficient in that it only updates reachable augmented states and ignores the unreachable ones, the policy that it finds is correct only for the given user-defined threshold θ_0. If the value of the threshold changes, then one has to recompute a new policy for the new threshold. In some risk-sensitive applications, the threshold might change during policy execution due to exogenous circumstances.

We thus introduce TVI-DP—an algorithm based on TVI and dynamic programming (DP) that finds optimal policies for all thresholds $\theta \in \mathbf{\Theta}$ that are bounded from above: $0 \leq \theta \leq \theta_0$. The algorithm requires the costs and thresholds to have a finite precision, which results in a finite set of the thresholds $\mathbf{\Theta}$. For example, if one limits the precision on the costs and thresholds to integers, as we do in this paper for the sake of clarity, then $|\mathbf{\Theta}| = \theta_0 + 1$.[4]

Algorithm 2 shows the pseudocode of the algorithm. Like TVI-DFS, TVI-DP also partitions the augmented MDP state space into SCCs with Tarjan's algorithm (line 31). However, unlike TVI-DFS, TVI-DP runs Tarjan's algorithm on the *transposed graph* from all goal states $s_g \in \mathbf{G}$, where all the direction of the transition edges are reversed.

Once all the SCCs are found, TVI-DP updates the augmented states starting from the augmented states with thresholds $\theta = 0$ to the states with thresholds $\theta = \theta_0$ (line 32). For each group of augmented states with the same threshold, TVI-DP updates the states in the SCCs in reverse topological sort order (lines 33-34). For each augmented state (s, θ) that is not in an SCC, TVI-DP updates its probability value and action to that in the augmented state $(s, \theta - 1)$ (lines 35-37). An important property here is that all augmented states in an SCC must have the same threshold (see Theorem 1). Therefore, TVI-DP ensures that before updating any augmented state, it updates all augmented states with smaller thresholds first.

Figures 2(b) and 2(c) show the resulting augmented MDP when one runs the TVI-DP algorithm on our example MDP with s_0 as the start state, s_4 as the goal state, and a maximum threshold $\theta_0 = 6$. The indices of the SCCs in reverse topological sort order are shown on the upper left corner of the rounded rectangles. Figure 2(b) shows the transitions of augmented states that are in SCCs. These transitions were computed by the UPDATE-SCC procedure. Figure 2(c) shows all the transitions including the transitions of augmented states that are not in SCCs. Table 1(b) shows the probability $P(s, \theta)$ of each reachable augmented state (s, θ).

Note that TVI-DP finds a larger number of reachable augmented states than TVI-DFS. The reason is that TVI-DP finds a policy for each possible combination of start state and starting threshold except for combinations of start states and starting thresholds with zero probability of reaching a goal with a cost within the threshold.

Similar to TVI-DFS, one can also optimize TVI-DP by integrating Tarjan's algorithm to find SCCs (line 31) with the procedure to update the augmented states (lines 34, 36-37). However, since TVI-DP calls Tarjan's algorithm on the transposed graph and starts from the goal states, its optimizations are slightly different than those in TVI-DFS. TVI-DP finds all the SCCs for a particular threshold θ by running Tarjan's algorithm on the subgraph consisting only of augmented states with that threshold, updates all the augmented states in that subgraph, and proceeds to find the SCCs in the next threshold $\theta + 1$. For example, TVI-DP finds SCCs 2 and 3 in Figure 2(c), updates all the augmented states with $\theta = 1$, before proceeding to find SCCs in the layer $\theta = 2$.

Finally, similar to TVI-DFS, if the cost function does not return zero costs, then one can also optimize TVI-DP to not use Tarjan's algorithm to find the SCCs and to not check for convergence in each SCC.

TVI-Bidirectional

Lastly, we also introduce an algorithm called TVI-Bidirectional that combines both TVI-DFS and TVI-DP. It searches top-down with TVI-DFS from a given start state and starting threshold (s_0, θ_0) and bottom-up from the goal states $s_g \in \mathbf{G}$ with TVI-DP. When the policies of the two algorithms intersect at the augmented states with thresholds $\theta = \theta_0/2$, the policies can be combined into a single policy.

Theoretical Results

Theorem 1 *In an RS-MDP, all augmented states in an SCC have the same cost threshold.*

[4]One can convert problems with higher precision costs and thresholds to problems with integer costs and thresholds by multiplying them with a sufficiently large constant.

PROOF SKETCH: We prove the theorem by showing that if two augmented states (s, θ) and $(\hat{s}, \hat{\theta})$ have different thresholds, then they must be in different SCCs. Assume that (s, θ) can transition to $(\hat{s}, \hat{\theta})$ via action a. Thus, $\hat{\theta} = \theta - C(s, a, \hat{s})$.

- If $C(s, a, \hat{s}) > 0$, then $\theta > \hat{\theta}$. Additionally, $(\hat{s}, \hat{\theta})$ cannot transition back to (s, θ) as the cost function C returns non-negative costs. As there are no transition cycles, the two augmented states cannot be in the same SCC.
- If $C(s, a, \hat{s}) = 0$, then $\theta = \hat{\theta}$. It is possible for $(\hat{s}, \hat{\theta})$ to transition back to (s, θ), in which case we have a cycle, and both augmented states are in the same SCC. ■

Lemma 1 *The* UPDATE-SCC *procedure is correct and complete given that the reachable probabilities of augmented states in downstream SCCs are correct.*

PROOF SKETCH: UPDATE-SCC is similar to the Bellman update for SCCs in TVI. We use the mechanism employed to prove correctness and completeness of Bellman update in guaranteeing correctness and completeness for the updates in Equation 7 for all the augmented states. ■

Theorem 2 *TVI-DFS is correct and complete.*

PROOF SKETCH: The reachable probability of an augmented state (s, θ) depends only on the reachable probability of its successors $(\hat{s}, \hat{\theta})$ (see Equation 7). Therefore, the reachable probability of augmented states in an SCC depend only on the reachable probability of augmented states in the same SCC and downstream SCCs. Since downstream SCCs are updated before upstream SCCs (the SCCs are updated in reverse topological sort order), the reachable probabilities are correct after the update (see Lemma 1). Thus, the algorithm is correct. The algorithm is also complete because each SCC is updated only once and each update is guaranteed to converge (see Lemma 1). ■

Theorem 3 *TVI-DP is correct and complete.*

PROOF SKETCH: For augmented states that are in SCCs, their reachable probabilities are correct for the same reason as that in TVI-DFS. For each augmented state (s, θ) that is not in an SCC, its path to an augmented goal state (s_g, θ_g) is the same path as that from $(s, \theta - 1)$ to $(s_g, \theta_g - 1)$, except that the threshold of each augmented state is added by 1. The reachable probability for each augmented state (s', θ') in that path is thus the same as that for $(s', \theta' - 1)$. Thus, copying the probabilities and policies (lines 36 and 37) is correct. Consequently, the algorithm is correct. The algorithm is also complete because each SCC is updated only once (see Lemma 1) and each augmented state that is not in an SCC has its probability updated only once. ■

Theorem 4 *In an RS-MDP, optimal policies are stationary and deterministic in the augmented state space.*

PROOF SKETCH: The augmented MDP is a MAXPROB MDP (Kolobov et al., 2011). In a MAXPROB MDP, an optimal policy is stationary and deterministic (Kolobov, 2013). Thus, optimal policies for RS-MDPs are also stationary and deterministic. ■

Theorem 5 *Solving RS-MDPs optimally is P-hard in the original state space.*

PROOF SKETCH: Similar to the proof in (Papadimitriou and Tsitsiklis, 1987), one can easily reduce a Circuit Value Problem to an RS-MDP. ■

Notice that solving RS-MDPs optimally is not P-complete because RS-MDPs are not in P in the original state space; solving them requires specifying an action for each augmented state in the set $\mathbf{S} \times \mathbf{\Theta}$, which, in turn, could be exponential in the size of $\mathbf{S}$ if $|\mathbf{\Theta}| = 2^{|\mathbf{S}|}$.

Related Work

While most of the MDP algorithms seek risk-neutral policies, that is, policies that minimize the expected cost or maximize the expected reward, there are several exceptions that seek risk-sensitive policies. As mentioned earlier, Yu, Lin, and Yan (1998) introduced RS-MDPs and introduced a VI-like algorithm to solve it.

McMillen and Veloso (2007) solved a specific type of RS-MDPs—finite-horizon RS-MDPs with zero-sum utility functions—inspired by robot soccer. They used a dynamic programming based algorithm that performs a one-sweep backup from the horizon to the starting time step to solve their problem.

Liu and Koenig (2005, 2006, 2008) generalized RS-MDPs by mapping the MDP rewards to risk-sensitive utility functions and sought to find policies that maximize the expected utility—an RS-MDP is a specific case, where the utility function is a step function. They introduced *Functional Value Iteration* (FVI), which finds optimal policies for any one-switch utility functions that are combinations of linear and exponential functions.

Ermon et al. (2012) extended their work by including a requirement that the returned policy needs to satisfy certain worst-case guarantees in addition to the expected utility maximization criterion. The worst-case guarantee is similar to those enforced by *Constrained MDPs* (Altman, 1999; Dolgov and Durfee, 2005), which enforces all constraints (e.g., the cumulative cost of a trajectory is no larger than a threshold) as hard constraints that cannot be violated. In contrast, RS-MDPs allow constraints to be violated but minimize the probability of that happening.

Kolobov et al. (2011) have also introduced an optimization criterion for MDPs, which is to maximize the probability of getting to a goal independent of cost. In order to solve for this criterion, they create a MAXPROB MDP that corresponds to the original MDP and solve that MAXPROB MDP. A MAXPROB MDP is thus equivalent to an RS-MDP where the objective is to find a policy π that maximizes $Pr(c^{\mathcal{T}(\pi)}(s_0) < \infty)$.

Defourny, Ernst, and Wehenkel (2008) introduced another criterion, where they seek to find a policy π that minimizes the expected cost *and* satisfies $Pr(c^{\mathcal{T}(\pi)}(s_0) \leq \theta_0) \geq p$, where p is a user-defined minimum probability threshold. This problem is similar to *Orienteering Problems* (OPs), which are also known as prize-collecting traveling salesman problems. In OPs, cities have associated rewards, and the

θ_0	$P(s_0, \theta_0)$	VI	FVI	TVI-DFS		TVI-DP		TVI-Bidirectional	
		time (s)	time (s)	time (s)	#SCCs	time (s)	#SCCs	time (s)	#SCCs
$0.25 \cdot \mathcal{C}^{\pi}(s_0)$	0.18	38.47	23.63	2.82	3,134,824	**2.14**	5,591,122	2.42	2,759,843
$0.50 \cdot \mathcal{C}^{\pi}(s_0)$	0.38	146.43	180.61	7.77	11,657,785	**4.76**	14,314,249	5.34	10,993,543
$0.75 \cdot \mathcal{C}^{\pi}(s_0)$	0.52	294.59	731.55	12.69	20,187,753	**7.32**	23,037,377	9.11	19,620,136
$1.00 \cdot \mathcal{C}^{\pi}(s_0)$	0.64	477.7	1307.34	17.62	28,717,721	**9.94**	31,760,504	12.91	28,246,639
$1.25 \cdot \mathcal{C}^{\pi}(s_0)$	0.72	696.05	2373.46	22.62	37,249,652	**12.61**	40,485,632	16.79	36,875,149
$1.50 \cdot \mathcal{C}^{\pi}(s_0)$	0.79	941.46	3934.78	27.63	45,781,574	**15.25**	49,210,759	20.58	45,503,650

Table 2: Results of the Randomly Generated MDPs with 1 Goal State

$\|\mathbf{G}\|$		$\theta_0 = 0.25 \cdot \mathcal{C}^{\pi}(s_0)$			$\theta_0 = 0.50 \cdot \mathcal{C}^{\pi}(s_0)$			$\theta_0 = 1.00 \cdot \mathcal{C}^{\pi}(s_0)$		
		1	10	100	1	10	100	1	10	100
TVI-DFS	time (ms)	2,823	**182**	**14**	7,774	2,050	**46**	17,620	6,319	**1,065**
	#SCCs	3,134,824	18,863	9	11,657,785	2,255,975	3,124	28,717,721	9,587,331	1,159,199
TVI-DP	time (ms)	**2,142**	722	171	**4,762**	**1,842**	482	**9,944**	**4,098**	1,218
	#SCCs	5,591,122	1,210,976	22,950	14,314,249	4,957,647	780,862	31,760,504	12,458,139	3,140,533

Table 3: Results of the Randomly Generated MDPs with Multiple Goal States

goal is to visit a subset of cities that maximizes the reward with the condition that the total traveling time is within a given maximum (Tsiligrides, 1984). *Stochastic OPs* (SOPs) extend OPs by assuming that the traveling times are stochastic (Campbell, Gendreau, and Thomas, 2011), and *Dynamic SOPs* extend SOPs by assuming that the stochastic traveling times are dynamic (time-dependent) (Lau et al., 2012; Varakantham and Kumar, 2013).

Experimental Results

We evaluate the TVI-DFS, TVI-DP, and TVI-Bidirectional algorithms against VI (on the augmented MDP)[5] and FVI. The VI algorithm is thus similar to that proposed by Yu, Lin, and Yan (1998). We run the algorithms on two sets of domains: (*i*) randomly generated MDPs, and (*ii*) ICAPS 2011 International Probabilistic Planning Competition (IPPC) domains. We conducted our experiments on a quad-core 3.40 GHz machine with 8GB of RAM.

Randomly Generated MDPs: We randomly generated MDPs with 10,000 states, 2 actions per state, and 2 successors per action. We randomly chose the costs from the range $[0, 100]$, varied the initial cost thresholds θ_0 according to minimum expected cost $\mathcal{C}^{\pi}(s_0)$, and varied the number of goal states from 1 to 100.

Tables 2 and 3 show our results.[6] We show the smallest runtime for each configuration in bold. We make the following observations:

- On problems with a single goal state, TVI-DP is faster than TVI-DFS. The reason is that the number of SCCs for both TVI-DFS and TVI-DP are about the same and the runtime overhead per SCC of TVI-DP is smaller than that of TVI-DFS. Both TVI-DFS and TVI-DP call Tarjan's algorithm recursively (similar to a recursive DFS function) to find SCCs. For TVI-DP, the largest number of recursive calls in memory is the largest number of SCCs in any subgraph consisting only of augmented states with the same threshold. This number is typically small. In contrast, for TVI-DFS, the largest number of recursive calls in memory is the height of the entire SCC transition tree, which can be large. The runtime overhead grows with the number of recursive calls in memory. TVI-DP thus has a smaller runtime overhead than TVI-DFS.
- On problems with a large number of goal states, despite the larger runtime overhead per SCC, TVI-DFS is still faster than TVI-DP. The reason is that TVI-DFS generates significantly fewer SCCs, often by more than one order of magnitude, than TVI-DP in these problems. As TVI-DP constructs the SCC transition tree on the transposed graph from each goal state, its number of SCCs grows with the number of goal states. In contrast, TVI-DFS constructs the SCC transition tree from each start state and, thus, its number of SCCs is less dependent on the number of goal states.
- The runtime of TVI-Bidirectional is in between the runtimes of TVI-DFS and TVI-DP in all cases. This result is to be expected since the algorithm is sped up by the faster algorithm but slowed down by the slower algorithm.
- All three TVI-based algorithms are faster than VI. The reason is similar to why TVI is faster than VI on regular MDPs—VI needs to update all augmented states in each iteration while TVI only needs to update the augmented states in a single SCC in each iteration.
- All three TVI-based algorithms are also faster than FVI. The reason is that they are designed to solve RS-MDPs specifically and, thus, exploits utility-dependent properties in RS-MDPs. In contrast, FVI is designed to solve general one-switch utility functions.
- Finally, as expected, the reachable probability of the augmented start state $P(s_0, \theta_0)$ increases with increasing

[5]We considered only the reachable augmented states in the augmented MDP.

[6]The runtime of VI does not include the runtime to generate the augmented MDP.

(a) $\theta_0 = 0.5 \cdot \mathcal{C}^\pi(s_0)$

Domain	$P(s_0, \theta_0)$	VI	FVI	TVI-DFS		TVI-DP		TVI-Bidirectional	
		time (ms)	time (ms)	time (ms)	#SCCs	time (ms)	#SCCs	time (ms)	#SCCs
Crossing Traffic	0.34	82	10,150	**19**	27,368	227	317,860	110	90,215
Elevators	0.10	20,228	24,294	**2,296**	1,293,481	3,184	1,772,491	3,439	1,377,087
Game of Life	0.00	2,755	42,503	771	160,430	**735**	175,249	785	182,398
Navigation	0.00	8	1,075	**4**	40	61	46,685	32	10,610
Reconnaissance	0.00	21	1,396	**20**	1	293	57,984	160	25,472
Skill Teaching	0.00	79	73,899	**74**	1	4,259	3,367,968	1,772	927,159
SysAdmin	0.05	120,639	637,868	**12,476**	608,039	225,318	624,315	110,084	832,023
Traffic	0.33	1,110	27,175	**101**	141,072	703	1,723,891	439	712,191
(b) $\theta_0 = 1.0 \cdot \mathcal{C}^\pi(s_0)$									
Crossing Traffic	0.69	248	16,902	**44**	76,087	452	754,248	252	347,868
Elevators	0.52	53,552	70,242	**6,961**	6,043,855	7,127	8,235,034	8,617	6,697,757
Game of Life	0.51	6,471	51,372	**1,328**	277,856	1,456	293,773	1,624	368,886
Navigation	0.00	40	1,507	**12**	2,630	126	120,520	70	47,935
Reconnaissance	0.77	45	4,485	**42**	3,876	588	123,264	322	59,796
Skill Teaching	0.58	3,269	152,618	**433**	216,308	9,069	8,085,337	4,400	3,370,019
SysAdmin	0.56	223,781	1,071,347	**26,921**	1,347,970	480,052	1,385,249	250,196	1,972,285
Traffic	0.59	2,010	45,949	**206**	284,229	1,492	4,661,085	829	1,893,678
(c) $\theta_0 = 1.5 \cdot \mathcal{C}^\pi(s_0)$									
Crossing Traffic	0.89	448	21,061	**73**	129,609	704	1,198,408	403	645,614
Elevators	0.89	86,351	109,579	11,849	11,670,296	**11,094**	15,851,101	13,544	13,624,710
Game of Life	1.00	6,391	52,979	**1,579**	332,161	2,138	349,138	2,438	539,662
Navigation	1.00	55	1,514	**16**	8,649	183	120,920	108	104,214
Reconnaissance	1.00	67	4,474	**61**	9,124	864	127,872	482	99,748
Skill Teaching	0.97	8,982	196,538	**1,008**	664,274	13,440	12,056,444	7,028	5,977,072
SysAdmin	0.91	283,092	1,445,475	**40,121**	2,033,750	732,474	2,092,030	388,462	3,045,182
Traffic	0.82	2,609	55,410	**302**	407,688	2,259	7,433,331	1,336	3,589,219

Table 4: Results of the ICAPS 2011 IPPC Domains

threshold θ_0.

Therefore, for problems with a small number of goal states, TVI-DP is better suited. For problems with a large number of goal states, TVI-DFS is better suited. Nonetheless, TVI-DP finds policies for all combinations of start states and initial cost thresholds that are no larger than a maximum threshold θ_0.

ICAPS 2011 IPPC Domains: We used the eight domains from the ICAPS 2011 International Probabilistic Planning Competition (IPPC) in our second set of experiments. For each domain, we report the results of the largest instance that fit in memory. The exceptions are as follows: (*i*) For the *Navigation* domain, all the instances were too small and could not sufficiently illustrate the difference in runtimes of the various algorithms. As such, we created a larger instance that uses the same domain logic and report the results of that instance. (*ii*) For the *Game of Life*, *Reconnaissance*, *SysAdmin*, and *Traffic* domains, all the instances were too large and could not fit in memory. Therefore, we created smaller instances that use the same domain logic and report the results of those instances.

Table 4 shows our results for three initial cost threshold θ_0 values, as a function of the minimum expected cost $\mathcal{C}^\pi(s_0)$. The trends in these results are similar to the trends in the randomly generated MDPs with large number of goal states. As these problems are finite-horizon problems, each augmented state is actually a state-timestep-threshold (s, t, θ) tuple, and each augmented state (s, H) with the horizon H as the time step is a goal state. Thus, the number of goal states is large.

Conclusions

While researchers have made significant progress on RS-MDPs, for example, by mapping MDP rewards to risk-sensitive utility functions and finding policies that maximize the expected utility (Liu and Koenig, 2005, 2006, 2008), little progress have been made on solving the original RS-MDP (with hard deadlines) since (Yu, Lin, and Yan, 1998).

In this paper, we revisit this problem and introduce new algorithms, TVI-DFS and TVI-DP, that combine TVI and traditional methods like depth-first search and dynamic programming, respectively. Our experimental results show that both TVI-DFS and TVI-DP are faster than VI and FVI. TVI-DP also has the added advantage that it finds policies for all combinations of start states and initial cost thresholds that are no larger than a maximum threshold θ_0. Thus, it is suitable in problems where the initial cost threshold is not known a priori and can be set at execution time. To the best of our knowledge, this is the first algorithm that is able to provide such policies.

References

Altman, E. 1999. *Constrained Markov Decision Processes*. Stochastic Modeling Series. Chapman and Hall/CRC.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Bertsekas, D. 2000. *Dynamic Programming and Optimal Control*. Athena Scientific.

Blythe, J. 1999. Decision-theoretic planning. *AI Magazine* 20(2):37–54.

Bonet, B., and Geffner, H. 2003. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1233–1238.

Campbell, A.; Gendreau, M.; and Thomas, B. 2011. The orienteering problem with stochastic travel and service times. *Annals of Operations Research* 186(1):61–81.

Cohen, P.; Greenberg, M.; Hart, D.; and Howe, A. 1989. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine* 10(3):32–48.

Dai, P.; Mausam; Weld, D.; and Goldsmith, J. 2011. Topological value iteration algorithms. *Journal of Artificial Intelligence* 42(1):181–209.

Defourny, B.; Ernst, D.; and Wehenkel, L. 2008. Risk-aware decision making and dynamic programming. In *NIPS 2008 Workshop on Model Uncertainty and Risk in Reinforcement Learning*.

Dolgov, D. A., and Durfee, E. H. 2005. Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1326–1331.

Ermon, S.; Conrad, J.; Gomes, C.; and Selman, B. 2011. Risk-sensitive policies for sustainable renewable resource allocation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1942–1948.

Ermon, S.; Gomes, C.; Selman, B.; and Vladimirsky, A. 2012. Probabilistic planning with non-linear utility functions and worst-case guarantees. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 965–972.

Goodwin, R.; Akkiraju, R.; and Wu, F. 2002. A decision-support system for quote generation. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence (IAAI)*, 830–837.

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic search for generalized stochastic shortest path MDPs. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 130–137.

Kolobov, A. 2013. *Scalable Methods and Expressive Models for Planning Under Uncertainty*. Ph.D. Dissertation, University of Washington.

Lau, H. C.; Yeoh, W.; Varakantham, P.; Nguyen, D. T.; and Chen, H. 2012. Dynamic stochastic orienteering problems for risk-aware applications. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 448–458.

Liu, Y., and Koenig, S. 2005. Risk-sensitive planning with one-switch utility functions: Value iteration. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 993–999.

Liu, Y., and Koenig, S. 2006. Functional value iteration for decision-theoretic planning with general utility functions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1186–1193.

Liu, Y., and Koenig, S. 2008. An exact algorithm for solving MDPs under risk-sensitive planning objectives with one-switch utility functions. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 453–460.

McMillen, C., and Veloso, M. 2007. Thresholded rewards: Acting optimally in timed, zero-sum games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 1250–1255.

Murthy, S.; Akkiraju, R.; Goodwin, R.; Keskinocak, P.; Rachlin, J.; Wu, F.; Yeh, J.; Fuhrer, R.; Aggarwal, S.; Sturzenbecker, M.; Jayaraman, R.; and Daigle, R. 1999. Cooperative multiobjective decision support for the paper industry. *Interfaces* 29(5):5–30.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Pell, B.; Bernard, D.; Chien, S.; Gat, E.; Muscettola, N.; Nayak, P. P.; Wagner, M.; and Williams, B. 1998. An autonomous spacecraft agent prototype. *Autonomous Robots* 5(1):29–52.

Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing* 1(2):146–160.

Tsiligrides, T. 1984. Heuristic methods applied to orienteering. *Journal of Operation Research Society* 35(9):797–809.

Varakantham, P., and Kumar, A. 2013. Optimization approaches for solving chance constrained stochastic orienteering problems. In *Proceedings of the International Conference on Algorithmic Decision Theory (ADT)*, 387–398.

Yu, S.; Lin, Y.; and Yan, P. 1998. Optimization models for the first arrival target distribution function in discrete time. *Journal of Mathematical Analysis and Applications* 225:193–223.

Zilberstein, S.; Washington, R.; Bernstein, D.; and Mouaddib, A.-I. 2002. Decision-theoretic control of planetary rovers. In *Revised Papers from the International Seminar on Advances in Plan-Based Control of Robotic Agents*, 270–289. Springer-Verlag.