Risk-Aware Stochastic Shortest Path

Tobias Meggendorfer

Institute of Science and Technology Austria, Am Campus 1, 3400 Klosterneuburg, Austria
tobias.meggendorfer@ist.ac.at

Abstract

We treat the problem of *risk-aware control* for *stochastic shortest path* (SSP) on *Markov decision processes* (MDP). Typically, expectation is considered for SSP, which however is oblivious to the incurred risk. We present an alternative view, instead optimizing *conditional value-at-risk* (CVaR), an established risk measure. We treat both Markov chains as well as MDP and introduce, through novel insights, two algorithms, based on linear programming and value iteration, respectively. Both algorithms offer precise and provably correct solutions. Evaluation of our prototype implementation shows that risk-aware control is feasible on several moderately sized models.

Introduction

Markov decision processes (MDP) are a standard model for sequential decision making in uncertain environments, applied in, for example, robot motion planning; see e.g. (White 1993, 1985) for a variety of further examples. Usually, one aims to control such a system optimally with respect to a "performance rating", called *objective*. In this work, we consider the *stochastic shortest path* (SSP) objective (Bertsekas and Tsitsiklis 1991), where the goal is to minimize the accumulated cost until a given set of target states is reached.

Traditionally, one seeks a policy minimizing the expectation of this accumulated cost. However, this policy willingly accepts arbitrary risks to achieve a minimal increase in expected profit. This may be undesirable, especially when the system in question, for example, models a situation which takes a long time to unfold or is only executed once, such as a retirement savings plan or Mars rover path planning. In particular, the law of large numbers is not applicable, and expectation alone provides little insight in the actual dynamics.

To remedy this issue, *risk-aware control* proposes several ideas. A popular approach is to quantify the risk incurred by a policy and then optimizing this *risk measure* instead of expectation. We briefly discuss some relevant measures: *Variance* does not focus on bad cases and may even incentivize intentionally performing suboptimally in unexpectedly good situations. *Worst case* analysis often is too pessimistic in probabilistic environments, considering events with probability 0 such as a fair coin toss never yielding heads. *Value-at-risk*
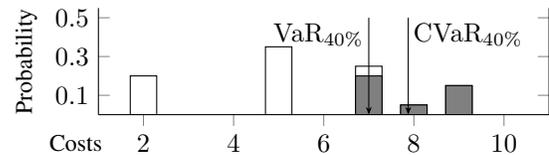
Figure 1: Example distribution over costs to showcase VaR and CVaR with threshold $\mathtt{t} = 40\%$. The bars represent the respective probabilities, while the grey area depicts the part considered by CVaR: The sum of the grey area equals the specified threshold of 40%, the expectation over it is 7.875.

(VaR) is the worst p-quantile for a given threshold $\mathtt{t} \in [0, 1]$. It approximates the notion of a "reasonably likely" bad case. However, VaR ignores the magnitude of worse cases, and has been characterized "seductive, but dangerous" and "not sufficient to control risk" (Beder 1995). *Conditional value-at-risk* (CVaR) (average value-at-risk, expected shortfall) yields the expectation over all outcomes worse than the VaR, i.e. the "tail loss"; see Fig. 1 for a sketch. As such, it considers outliers, weighted accordingly. It is an established and "more consistent measure of risk" (Artzner et al. 1999; Rockafellar, Uryasev et al. 2000), gaining popularity in various fields. We direct the interested reader to (Sarykalin, Serraino, and Uryasev 2008) for detailed comparison between VaR and CVaR, (Filippi, Guastaroba, and Speranza 2020) for a survey of CVaR applications, and (Shapiro, Dentcheva, and Ruszczynski 2014) for further risk measures.

Motivated by these observations, our primary goal in this work is to provide risk-aware control for SSP objectives through the optimization of its CVaR.

Related Work　Firstly, (Křetínský and Meggendorfer 2018) also considers CVaR, however for mean payoff instead of SSP and only provides an linear programming based solution, while we additionally present a value iteration approach. Secondly, (Chow et al. 2015) treats the *discounted* variant of the problem, fundamentally relying on the discounting factor to bound the error of the approximation. Moreover, (Carpin, Chow, and Pavone 2016) considers two variants of the problem. For the first, much more restricted variant, they present an approximative formulation together with an algorithm. For the second variant, which is more general and closer to

our problem, they present an approximation of the previous approximative formulation, without any guaranteed bounds. In contrast to the former two, our approach yields a provably correct, precise result for the infinite horizon problem on MDP, only using standard assumptions.

Many other works dealing with CVaR on MDP, e.g. (Borkar and Jain 2014; Keramati et al. 2020; Bäuerle and Ott 2011), often consider finite horizon and/or discounted costs, but not the undiscounted infinite horizon variant. A broad spectrum of research focusses on risk-aware *reinforcement learning*, typically aiming at best-effort solutions converging to the optimal value in the limit at most and without any guarantees. Note that especially when considering risk, providing reliable guarantees may be considered vital. Often, these best-effort solutions also introduce additional constraints, such as restricting to a suboptimal class of policies. A different perspective considers *time consistent* risk measures (Ruszczynski 2010), where risk effectively "accumulates" along the run; see (Tamar et al. 2017) for a CVaR-variant.

Contributions & Novelty We treat, to our knowledge for the first time, the problem of optimizing the *global, infinite horizon* risk of SSP on MDP through CVaR, providing *provably correct results*. We first discuss the problem on Markov chains, derive a central *characterization of CVaR for SSP*, and provide a tailored algorithm. Then, we present *two novel solution approaches* for MDP, based on this characterization. One is based on linear programming and one on value iteration (both exponential). While the primary focus of this work is the theoretical contribution, we also evaluate a prototype implementation, showing that risk-aware control with guarantees is practical even for infinite horizon problems.

Due to space constraints, most proofs have been moved to the technical report (Meggendorfer 2022).

Preliminaries

A (finite, discrete time, time-homogeneous) *Markov decision process* (MDP) (Puterman 1994; Bertsekas 2005) is a tuple $\mathcal{M} = (S, A, \hat{s}, \Delta, C)$, where S is a finite set of *states*, A is a finite set of *actions*, $\hat{s} \in S$ is the *initial state*, $\Delta(s, a, s') = \Pr[s'|s, a]$ is the Markovian *transition function*, and $C(s, a) \in \mathbb{N}_0$ is the non-negative, integer *cost* associated with taking action a in state s. We choose integer costs for simplicity, however our methods are also applicable to rational costs (by rescaling). An action a is *available* in state s if $\sum_{s' \in S} \Delta(s, a, s') = 1$ (the sum is 0 otherwise). We write $A(s) \subseteq A$ for the set of all actions available in state s.

A *Markov chain* (MC) is an MDP where $|A(s)| = 1$ for all states $s \in S$, i.e. the system is fully probabilistic.

The non-determinism in MDP is resolved by *policies*, mappings from finite paths to distributions over actions. The set of all policies is denoted by Π. A policy is called (i) *deterministic* if it always yields a unique action, (ii) *memoryless* (or *stationary*) if it only depends on the current state, and (iii) *Markovian* if it only depends on the number of steps already performed. Technically, an MDP with a policy *induces* a Markov chain, which allows to reason about the now fully probabilistic system. See e.g. (Puterman 1994, Chp. 2) or (Baier and Katoen 2008, Sec. 10.6) for formal details.

Stochastic Shortest Path (SSP) (Bertsekas and Tsitsiklis 1991) is a common objective on MDP, specified by an MDP and a set of *goal states* $\mathcal{G} \subseteq S$. We are interested in the total accumulated cost until a goal state is reached. We write $\mathcal{R}^{s,\pi}$ to denote the distribution over total costs achieved by policy π starting in state s. Typically, one optimizes the *expected* total cost, i.e. given a state s, find a policy $\pi \in \Pi$ such that

$$V^\pi(s) := \mathbb{E}\left[\sum_{t=0}^\infty C(s_t, a_t) \mid s, \pi\right] = \mathbb{E}[\mathcal{R}^{s,\pi}].$$

is minimal. As already suggested, instead of expectation, we however are interested in optimizing a *risk measure* of $\mathcal{R}^{s,\pi}$.

Conditional Value-at-Risk (CVaR) (also known as Average Value-at-Risk (AVaR)) is our proposed alternative to expectation. To introduce CVaR, we first need to define the notion of *value-at-risk* (VaR). Intuitively, VaR tries to answer the question "what is a reasonable bad outcome?" VaR is parametrized by a threshold $\mathsf{t} \in [0, 1]$ and yields the *worst* t-*quantile*, i.e. a value v such that an outcome is worse than v with probability t. For example, the 50%-VaR effectively is the median. Formally, given a distribution over natural numbers $X : \mathbb{N}_0 \to [0, 1]$ and $\mathsf{t} < 1$ we define

$$\mathrm{VaR}_\mathsf{t}(X) := \min\{v \in \mathbb{N}_0 \mid \sum_{x=v+1}^\infty X(x) \le \mathsf{t}\}.$$

(As we are considering costs, larger values are worse.) For consistency, let $\mathrm{VaR}_1(X) := \min\{v \in \mathbb{N}_0 \mid X(v) > 0\}$. Note that for $\mathsf{t} = 0$ we may have $\mathrm{VaR}_0(X) = \infty$.

Example 1. Consider the distribution from Fig. 1, i.e. $X = \{2 \mapsto 20\%, 5 \mapsto 35\%, 7 \mapsto 25\%, 8 \mapsto 5\%, 9 \mapsto 15\%\}$. We see that $\mathrm{VaR}_{40\%}(X) = 7$. Maybe unexpectedly, we have $\mathrm{VaR}_{45\%}(X) = 5$ instead of 7, as $\Pr[X > 5]$ is exactly 45%. It is a matter of preference how to define this boundary case, and either works in our setting. In particular, it does not influence the definition of CVaR.

CVaR, also parametrized by a threshold $\mathsf{t} \in [0, 1]$, aims to answers the question "what can we expect from an average bad case?" Formally, CVaR equals the expectation of X conditional on only considering the worst t outcomes. Similar to Example 1, we need to apply special care when working with discrete distributions: Again recall the example from Fig. 1 with $\mathsf{t} = 40\%$. There, only 20% of the $X(7) = 25\%$ should be considered. Thus, CVaR is defined as follow. For a distribution X and threshold $\mathsf{t} > 0$, define $v := \mathrm{VaR}_\mathsf{t}(X)$ and $\mathfrak{V} := X > v$ the event of an outcome being strictly worse than the VaR. Then

$$\mathrm{CVaR}_\mathsf{t}(X) := \tfrac{1}{\mathsf{t}}\big(\Pr[\mathfrak{V}] \cdot \mathbb{E}[X \mid \mathfrak{V}] + (\mathsf{t} - \Pr[\mathfrak{V}]) \cdot v\big). \quad (1)$$

For the degenerate case of $\mathsf{t} = 0$, we define $\mathrm{CVaR}_0(X) := \lim_{\mathsf{t} \to 0} \mathrm{CVaR}_\mathsf{t}(X) = \mathrm{VaR}_0(X)$.

Remark 1. Observe that $\mathrm{CVaR}_0(X)$ is the worst-case of X and $\mathrm{CVaR}_1(X) = \mathbb{E}[X]$ the expectation of X; changing t thus *smoothly interpolates* between these extremes. As both of these extremal cases are already solved for SSP, we exclude them, i.e. assume $0 < \mathsf{t} < 1$.

See e.g. (Křetínský and Meggendorfer 2018, Sec. 3) for a more detailed discussion of CVaR on discrete distributions.

Problem Statement Together, the central question considered in this work is:

Given an SSP problem, what is the optimal CVaR?

Formally, given an MDP $\mathcal{M}$, cost function C, and threshold $0 < \mathsf{t} < 1$, we want to determine

$$\mathrm{CVaR}^*_{\mathsf{t}} := \inf_{\pi \in \Pi} \mathrm{CVaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s},\pi}).$$

We refer to this problem as **CVaR-SSP**.

Linear Programming (LP) (see e.g. (Schrijver 1999)) is an established problem solving technique with strong connections to MDP; many popular objectives allow for a natural LP formulation. An LP is characterized by a linear *objective function* f and a set of *linear inequality constraints* on its variables. The task then is to find the maximal (or minimal) value of f subject to the imposed constraints. This value can be computed in polynomial time (Khachiyan 1979; Karmarkar 1984). As such, LP is a popular tool to prove complexity bounds of many problems.

Value Iteration (VI) (Bellman 1966) is a popular practical approach to solve various questions related to MC and MDP, among others. As the name suggests, one repeatedly applies an iteration operator to a value vector v_i (typically one real value per state). For example, the canonical value iteration for SSP starts with $v_0(s) = 0$ for all $s \in S$ and then iterates

$$v_{i+1}(s) = \min_{a \in A(s)} C(s,a) + \sum_{s' \in S} \Delta(s,a,s') \cdot v_i(s').$$

Under the mentioned assumptions, this iteration converges to the true value in the limit, with an exponential worst-case bound to reach a given precision. In practice, VI typically performs very well, quite often outperforming LP approaches by a large margin. A similar trend emerges for our approaches.

Assumptions Finally, we introduce several standard assumptions for **CVaR-SSP**. First, we assume that $\hat{s} \notin \mathcal{G}$ (otherwise the problem would be trivial) and that all goal states are absorbing. Next follow two standard assumptions for SSP (Bertsekas and Tsitsiklis 1996). A policy is *proper* if the probability of eventually reaching the goal from every state is 1. We assume that (i) there exists a proper policy and (ii) for every improper policy π, $V^\pi(s)$ is infinite for at least one state s. Finally, we assume that the cost of an action is 0 *if and only if* the corresponding state is a goal state. This assumption, also used in e.g. (Bonet 2007; Carpin, Chow, and Pavone 2016), is mainly introduced for simplicity, we briefly discuss later on how it can be lifted. Note that (ii) follows from (i) and the latter assumption.

Reachability & Uniform Costs

We restrict to a simpler setting to explain central insights more clearly. Namely, we assume that costs are uniform, i.e. $C(s,a) = 1$ for all non-goal states. The total cost $\mathcal{R}^{s,\pi}$ now can be interpreted as "starting from state s with policy π, how many steps are needed to reach the goal?" The VaR corresponds to the first step after which a fraction of at least $1 - \mathsf{t}$ of all executions (abbreviated by "$1 - \mathsf{t}$ executions" in the following) have reached a goal state; CVaR is the overall expected number of steps to reach the goal for the remaining t executions. We discuss the general case afterwards.

Markov Chains

To get started, we first consider Markov chains. Since MC are purely stochastic, our problem changes from optimization to computation. For readability, we thus omit policies from superscripts such as $\mathcal{R}^{s,\pi}$ and write $\mathcal{R}^s$ instead.

Recall that VaR is the first time step after which $1 - \mathsf{t}$ executions have reached the goal. We can compute this step by iterating the transition relation of the MC, i.e. computing where the system is after n steps. This naturally also gives us the distribution of the remaining executions which have not yet reached the goal. To obtain the CVaR, we then need to consider the expected cost to reach the goal for this remainder, i.e. the classical SSP value.

Formally, fix an MC M and goal states $\mathcal{G}$. Let $p_n(s) := \Pr[s_n = s \mid \hat{s}]$ the probability that the system is in state s after n steps and $e(s) := \mathbb{E}[\mathcal{R}^s]$ the expected number of steps to reach a goal state starting in s. We define $\mathcal{N}_n := 1 - \sum_{s \in \mathcal{G}} p_n(s)$ the probability of not having reached the goal state after n steps. Then, $\mathrm{VaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s}})$ is the unique value n such that $\mathcal{N}_{n+1} < \mathsf{t} \leq \mathcal{N}_n$. By our assumptions, we have that $\mathcal{N}_n \to 0$ for $n \to \infty$, consequently such an n exists for every $\mathsf{t} > 0$. Finally, let $\mathcal{E}_n := \sum_{s \in S} p_n(s) \cdot e(s)$ the expected time to reach the goal after n steps. Note that we can include goal states in the sum as $e(s) = 0$ for all goal states and they are absorbing. Moreover, $\mathbb{E}[\mathcal{R}^s \mid \mathcal{R}^s > n] = n + \frac{1}{\mathcal{N}_n}\mathcal{E}_n$: We deliberately define $\mathcal{E}_n$ independent of the fraction of runs which already have reached the goal, thus the conditioning of CVaR requires re-weighting.

Together, we obtain an intuitive characterization of CVaR for SSP, which is the foundation for our solution approaches.

Theorem 1. *For* $\mathrm{VaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s}}) = n$, *we have*

$$\mathrm{CVaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s}}) = n + \tfrac{1}{\mathsf{t}}\mathcal{E}_n.$$

Proof. Inserting the above definitions in Eq. (1) yields

$$\mathrm{CVaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s}}) = \tfrac{1}{\mathsf{t}}\left(\mathcal{N}_n \cdot (n + \tfrac{1}{\mathcal{N}_n} \cdot \mathcal{E}_n) + (\mathsf{t} - \mathcal{N}_n) \cdot n\right)$$
$$= n + \tfrac{1}{\mathsf{t}}\mathcal{E}_n. \qquad \square$$

This already yields an effective algorithm for MC: We compute e using standard methods, iteratively compute p_n for increasing n to obtain $\mathrm{VaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s}})$, and together get $\mathrm{CVaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s}})$. Unfortunately, VaR may be of exponential size.

Lemma 1. *For every Markov chain M we have that* $\mathrm{VaR}_{\mathsf{t}}(\mathcal{R}^{\hat{s}}) \in \mathcal{O}(-\log \mathsf{t} \cdot |S| \cdot p_{\min}^{-|S|})$, *where $p_{\min}$ is the minimal transition probability in* M. *This bound is tight.*

Thus, our algorithm is exponential. However, for polynomial VaR the overall algorithm is polynomial, too, since e can be computed in polynomial time. For practical purposes, we can additionally exploit that $\mathcal{N}_n$ is monotone in n and employ binary search together with exponentiation by squaring.

Lemma 2. *On Markov chains, **CVaR-SSP** can be solved using polynomially many arithmetic operations.*

Note that the overall runtime of this algorithm still is exponential, since multiplication itself is not a constant time operation. In practice, our algorithm can benefit from efficient matrix-multiplication methods and fixed-point arithmetic.
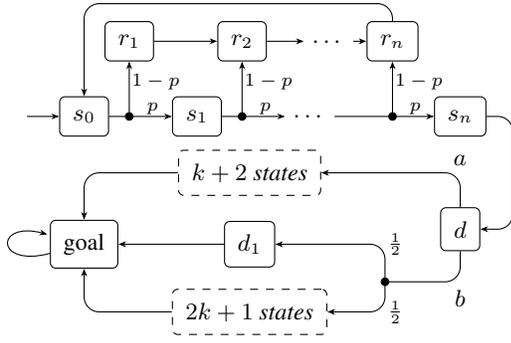
Figure 2: Exponential memory may be required. The upper part ensures that after $n + 1$ steps, a fraction of p^n executions are in s_n and the remaining $1 - p^n$ are in s_0. The lower part then comprises a choice between a safe option (action a) and a more risky but slightly more efficient option (action b).

Markov Decision Processes

Now, we move our focus from MC to MDP. We can re-use some of the observations from the previous section, however the addition of non-determinism complicates the problem significantly. In particular, there is no unique distribution p_n, it rather depends on the chosen policy π, which we denote by p_n^π. Analogously, we write $\mathcal{N}_n^\pi$ and $\mathcal{E}_n^\pi$ for the respective values achieved by a given policy π and abbreviate $\mathrm{CVaR_t}(\pi) := \mathrm{CVaR_t}(\mathcal{R}^{\hat{s},\pi})$ (analogous for $\mathrm{VaR_t}(\pi)$). Finally, we write $e(s)$ for the optimal expected cost to reach the target starting in s, i.e. the classical SSP value.

Note that we may have $\mathrm{VaR_t}(\pi) = \mathrm{CVaR_t}(\pi) = \infty$ for some π. However, under every *proper* policy π^p the goal is reached within finite time with probability 1. Thus $\mathrm{VaR_t}(\pi^p)$ is bounded, Theorem 1 remains applicable (by considering the induced MC), and $\mathrm{CVaR_t}(\pi^p) = n + \frac{1}{t}\mathcal{E}_n^{\pi^p} < \infty$ for $n = \mathrm{VaR_t}(\pi^p)$. Consequently, the optimal value is finite.

Before diving deeper into the solution concepts, we first prove that an optimal policy always exists.

Theorem 2. *We have* $\mathrm{CVaR_t^*} = \min_{\pi \in \Pi} \mathrm{CVaR_t}(\pi)$.

As a naive approach, one thus could try to enumerate all possible policies and apply the reasoning of the previous section. However, even when only considering memoryless deterministic policies, there may be exponentially many distributions p_n^π. Even worse, the following example shows that optimal policies may require *exponential memory*. This suggests that enumeration approaches such as policy iteration (another popular approach to solve problems on MDP), or a simple value iteration likely are bound to fail, since both of them typically work with local, "memoryless" values.

Example 2. Consider the MDP in Fig. 2 for any $n > 2k + 1$. In this case, every $i \cdot (n+1)$ steps, a "packet" of $(1 - p^n)^i \cdot p^n$ executions arrives at d. The optimal choice in d depends on the fraction of executions which are still in the upper part. If more than $1 - t$ are still "on top", the choice does not matter, since the current execution will surely reach the goal before the VaR, i.e. the packet is composed completely of "good" outcomes and not considered for CVaR. If more than $1 - t$ executions already are in the lower part beyond d, the
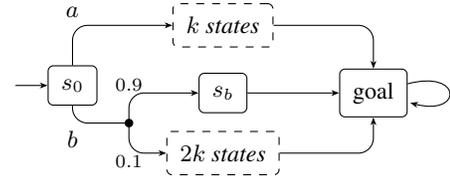
optimal choice is b, since the current packet only contains "bad" outcomes; only the expectation counts. However, for some i^*, the current packet contains exactly those executions which are at the $1 - t$ boundary, i.e. containing both good and bad ones. Then (for appropriate n and p) the optimal choice is action a. For example, if the current packet is composed of exactly 50% good and 50% bad, the expected performance of the bad fraction under a is $k + 2$ compared to $2k + 1$ under b. One can directly show that the step corresponding to i^* is exponential and thus the policy requires as much memory.

Remark 2. The example above shows that exponential memory is required. However, it does not prove that randomization is needed, and we have not found an example where this would be the case. We conjecture that deterministic policies actually are sufficient and leave this question for future work.

Despite the exponential memory requirement, we are able to derive two practical solution techniques, which we explain in the following. First, we again observe that once the VaR is reached, i.e. the system has performed $\mathrm{VaR_t}(\pi) = n$ steps, we are only interested in the expectation: Exactly those executions which have not reached the goal after n steps are considered in the expectation computation of CVaR.

Lemma 3. *Fix a policy π and let $n = \mathrm{VaR_t}(\pi)$. Then, there exists a policy π' which is stationary after n steps and* $\mathrm{CVaR_t}(\pi') \leq \mathrm{CVaR_t}(\pi)$.

So, intuitively, as before in the Markov chain case, after n steps we are only interested in the optimal expected time $e(s)$. We can use standard methods to compute this value (and a witness policy) in polynomial time (Puterman 1994, Chp. 7). This suggests that our task decomposes into two sub-problems, namely (i) reaching the goal quickly, resulting in a small VaR, and (ii) optimally distributing the remaining executions, i.e. states with a small expected time to reach the goal $e(s)$. One might feel tempted to first minimize the VaR and then, among "VaR-optimal" policies, choose one with optimal expected value on the remaining t executions. However, trying to achieve a VaR as small as possible at all costs may actually come with a significantly larger "tail" of the distribution – which is one of the main reasons why VaR is "seductive, but dangerous".

Example 3. Consider the MDP in Fig. 3 with $t = 0.15$. Action b is strictly preferred both for expectation as well as VaR optimization, while action a is CVaR-optimal.

More strikingly, observe that action b in the example is VaR optimal even if instead of $2k$ states there would be *arbitrarily many*. So, in a sense, VaR does not consider the t worst outcomes but rather the $1 - t$ best. By optimizing the first

Figure 3: VaR optimization is suboptimal.

$$\min \sum_{s \in S} p_{s,n} \cdot e(s) \text{ subject to}$$

All variables non-negative

$$p_{\hat{s},0} = 1 \qquad p_{s,0} = 0 \quad \forall s \in S, s \neq \hat{s}$$

$$p_{s,i} = \sum_{a \in A(s)} p_{s,a,i} \quad \forall s \in S, i < n$$

$$p_{s',i+1} = \sum_{s \in S, a \in A(s)} p_{s,a,i} \cdot \Delta(s, a, s') \quad \forall s' \in S, i < n$$

$$\sum_{s \in \mathcal{G}} p_{s,n-1} \leq 1 - \mathsf{t} \leq \sum_{s \in \mathcal{G}} p_{s,n}$$

Figure 4: LP to compute CVaR_t given VaR guess n.

$1 - \mathsf{t}$ executions, the remaining portion may be positioned disproportionately bad. Instead, we have to balance between issues (i) and (ii), and search for a *trade-off* between a small VaR and the distribution of the remaining executions. We present two different approaches to find this trade-off, based on linear programming and value iteration, respectively.

Linear Programming

For our linear programming approach, first assume that we are magically given the optimal VaR n, i.e. the VaR which allows us to obtain the optimal CVaR. Inspired by (Chatterjee, Kretínská, and Kretínský 2017, Fig. 3), we can construct an LP of size linear in n, where the set of solutions corresponds to all policies achieving this VaR, shown in Fig. 4. This LP then computes the minimal CVaR over these policies. Intuitively, the LP is obtained by *unrolling* the MDP until step n and building reachability constraints on that MDP.

Theorem 3. *If there exists a policy π such that $\mathrm{VaR}_\mathsf{t}(\pi) = n$ and $\mathrm{CVaR}_\mathsf{t}(\pi) = C$, the LP in Fig. 4 has a solution with value at most $\mathsf{t} \cdot (C - n)$. If the LP in Fig. 4 has a solution for some n with value E, there exists a policy achieving $\mathrm{CVaR}_\mathsf{t}(\pi) = n + \frac{1}{\mathsf{t}}E$.*

Proof sketch (see report). First part: We construct an assignment to the LP's variables: Set $p_{s,i} = p_i^\pi(s)$ and $p_{s,a,i} = \Pr[s_i = s, a_i = a \mid \hat{s}, \pi]$. This assignment satisfies the first three constraints of the LP. For the fourth constraint, observe that by $\mathrm{VaR}_\mathsf{t}(\pi) = n$, we have that $\sum_{s \in \mathcal{G}} p_{n-1}^\pi(s) < 1 - \mathsf{t} \leq \sum_{s \in \mathcal{G}} p_n^\pi(s)$. By Theorem 1, we have that $C = n + \frac{1}{\mathsf{t}}\mathcal{E}_n^\pi$. Since $\mathcal{E}_n^\pi = \sum_{s \in S} p_n^\pi(s) \cdot e(s) = \sum_{s \in S} p_{s,n} \cdot e(s)$, we get that $\mathcal{E}_n^\pi = \mathsf{t} \cdot (c - n)$, proving the claim.

Second part: We construct the policy π as follows. For the first n steps, at step i in state s, choose action a with probability $p_{s,a,i}$. Afterwards, i.e. starting from step n, in state s follow a policy achieving the optimal expected cost $e(s)$ (note the similarity to Lemma 3). Clearly, $p_i^\pi(s) = p_{s,i}$ and thus $v = \mathcal{E}_n^\pi$. We have $\mathrm{VaR}_\mathsf{t}(\pi) = n - 1$ if $\sum_{s \in \mathcal{G}} p_{s,n-1} = 1 - \mathsf{t}$ and $\mathrm{VaR}_\mathsf{t}(\pi) = n$ otherwise. In both cases, we can prove $\mathcal{E}_{n-1}^\pi = \mathsf{t} + \mathcal{E}_n^\pi$. Inserting yields $\mathrm{CVaR}_\mathsf{t}(\pi) = (n-1) + \frac{1}{\mathsf{t}}\mathcal{E}_{n-1}^\pi = n + \frac{1}{\mathsf{t}}\mathcal{E}_n^\pi = n + \frac{1}{\mathsf{t}}v.$ $\square$

This directly suggests an algorithm: We simply try each possible VaR value and solve the associated LP. Unfortunately, as suggested by Lemma 1, the VaR obtained by CVaR optimal policies may be at least exponential. We prove a matching upper bound in the following.

Lemma 4. *Fix a policy π and let π^* be an optimal policy. Then $\mathrm{VaR}_\mathsf{t}(\pi^*) \leq \mathrm{CVaR}_\mathsf{t}(\pi)$.*

Note that the VaR of CVaR optimal policies may not be optimal, i.e. potentially $\mathrm{VaR}_\mathsf{t}(\pi^*) > \mathrm{VaR}_\mathsf{t}(\pi)$, see Example 3.

Furthermore, by employing our assumption that a proper policy exists and combining it with Lemma 1, we can find a policy with CVaR of bounded size.

Lemma 5. *There is a policy with at most exponential CVaR.*

Lemmas 4 and 5 together then yield the desired result.

Corollary 1. $\mathrm{VaR}_\mathsf{t}(\pi^*)$ *is at most exponential.*

This implies that our LP algorithm is EXPTIME: We solve exponentially many linear programs of exponential size. However, Lemma 4 also yields a dynamic "stopping criterion" for our algorithm: We do not always need to try out all exponentially many possible values for VaR. Instead, once we found a solution, we can use the CVaR obtained by this solution as new upper bound for the VaR guesses. Thus, the exponential time solely depends on the magnitude of $\mathrm{CVaR}_\mathsf{t}(\pi^*)$. In particular, if $\mathrm{CVaR}_\mathsf{t}(\pi^*)$ is of polynomial size, our algorithm is PTIME, since we can stop after polynomially many steps.

We conclude this section with a series of remarks, putting our results into context.

Remark 3. We conjecture that these results are optimal, i.e. that **CVaR-SSP** is EXPTIME-complete. However, a proof of hardness seems surprisingly difficult: Only recently, (Balaji et al. 2019) proved that the (conceptually much simpler) problem of finite-horizon reachability is EXPTIME-complete; a question open for several decades, posed already by (Papadimitriou and Tsitsiklis 1987). Observe that deciding whether a policy π exists such that $\mathrm{VaR}_\mathsf{t}(\pi) \leq n$ is a special case of finite-horizon reachability. Unfortunately, the techniques of (Balaji et al. 2019) are not applicable to our case, since we assumed the existence of proper policies. We conjecture that the associated VaR problem nevertheless is EXPTIME-complete, too. Yet, even proving hardness of the VaR problem does *not* immediately prove that **CVaR-SSP** itself is EXPTIME-complete: There might be an algorithm which can determine the CVaR without explicitly determining the VaR. It however seems unlikely that CVaR can be accurately computed without knowledge of VaR.

Remark 4. In (Křetínský and Meggendorfer 2018), the authors also employed a "VaR-guessing" approach for a similar problem. In their setting however, only linearly many possible value for VaR exist, thus yielding a polynomial algorithm. See (Piribauer 2021, Thm. 3.33) for a translation of our case to the scenario of (Křetínský and Meggendorfer 2018) and an alternative proof for the exponential upper bound.

Remark 5. Our LP approach can be adapted to the constrained variant, i.e. answer the question "given that CVaR should be at least x, what is the maximal expectation?", by changing the CVaR objective to an appropriate constraint and adding expectation maximization as objective.

Value Iteration

While LP is appealing in theory due to its polynomial complexity, in practice it often is outperformed by approaches such as VI, despite worse theoretical complexity. We now discuss several insights which ultimately lead to a VI algorithm for our problem, yielding precise results. This is particularly intriguing since a VI approach for the similar scenario of (Křetínský and Meggendorfer 2018) remains elusive.

To derive a VI approach, we require an iteration operation which yields a value for each state based on the values of their respective successors. So, suppose we naively want to compute $\mathrm{CVaR_t}$ of a state s based on the $\mathrm{CVaR_t}$ of its successors. Unfortunately, we cannot simply combine the $\mathrm{CVaR_t}$ of the successors: For example, it might be the case that all "bad" outcomes (i.e. those worse than the $\mathrm{VaR_t}$) are all those which move to one particular successor. Hence, we would need $\mathrm{CVaR_1}$ for that successor and $\mathrm{CVaR_0}$ for all others. Consequently, we need to employ a different approach.

By definition of $\mathrm{CVaR_t}$, precisely a fraction of t executions starting in s are bad ones, and these have to distribute *somehow* over the successors. Thus, there is a weighting of successors, reflecting how the bad executions distribute.

Lemma 6. *Let s be a state, $a \in A(s)$ an available action, and π a policy. There exist weights $w : S \to [0, 1]$ such that $\sum_{s' \in S} \Delta(s, a, s') \cdot w(s') = \mathsf{t}$ and*

$$\mathrm{CVaR_t}(\mathcal{R}^{s,\pi}) =$$
$$1 + \sum_{a \in A(s), s' \in S} \pi((s), a) \Delta(s, a, s') \mathrm{CVaR}_{w(s')}(\mathcal{R}^{s', \pi^a}),$$

where (s) denotes a path comprising only s and π^a denotes the policy π after taking action a.

Proof. Follows directly from the above discussion. $\square$

Note the similarity of the above equation to a Bellman update: For known/fixed weights w, we would obtain a regular value iteration. Unfortunately, these weights globally depend on the policy; even the decision in a state s' which is not reachable from state s influences the weight distribution in s. Nevertheless, we can use the underlying insights to derive a value iteration approach.

Characterization through Pareto Sets

In light of Lemma 6, we do not want to compute the CVaR for a single threshold t, but for *all* thresholds $0 < \mathsf{t} \leq 1$. So, essentially, we aim to answer the question "given a threshold of t, what is the best achievable CVaR?" for all thresholds and all states. We approach this question with a new, novel perspective. In particular, we propose to answer a different question, namely, "for an arbitrary step-bound n, given that at least $1 - \mathsf{t}$ executions have to reach the goal within n steps, what is the best expected time to reach the goal after n steps?", a trade-off which can be described by a *Pareto set*. Quite surprisingly, we can both (i) derive CVaR from such a set and (ii) compute these sets for increasing n using VI.

Definition 1. Let $s \in S$ a state and $n \in \mathbb{N}$ a step bound. We define the *SSP Pareto set* $\mathfrak{P}_n^s \subseteq [0, 1] \times \mathbb{R}_{\geq 0}$ where

$(p, E) \in \mathfrak{P}_n^s$ iff there exists a policy π such that $\mathcal{N}_n^\pi \leq 1 - p$ and $\mathcal{E}_n^\pi \leq E$, i.e. (i) the probability to reach a goal state within n steps starting in s is at least p, and (ii) after n steps, the expected time to reach goal states is at most E.

Lemma 7. *For every $(1 - \mathsf{t}, E) \in \mathfrak{P}_n^{\hat{s}}$ with witness policy π, we have $\mathrm{CVaR_t}(\pi) \leq n + \frac{1}{\mathsf{t}} E$. For every policy π, we have $(1 - \mathsf{t}, \mathsf{t} \cdot (\mathrm{CVaR_t}(\pi) - n)) \in \mathfrak{P}_n^{\hat{s}}$ where $n = \mathrm{VaR_t}(\pi)$.*

To obtain an algorithm based on this idea, we need an effective procedure to compute $\mathfrak{P}_n^s$. To this end, we show that $\mathfrak{P}_n^s$ is a convex polygon where vertices correspond to deterministic policies, and show how $\mathfrak{P}_n^s$ can be computed using a Bellman-style iteration.

Lemma 8. *The set $\mathfrak{P}_n^s$ is an upward and leftward closed, convex polygon where all vertices correspond to Markovian deterministic policies.*

Proof sketch (see report). Closure and Convexity follow directly. We prove the polygon-claim by induction on n.

For $n = 0$, we either have that $\mathfrak{P}_0^s = [0, 1] \times \mathbb{R}_{\geq 0}$ if $s \in \mathcal{G}$, or, if $s \notin \mathcal{G}$, $\mathfrak{P}_0^s = \{0\} \times [e(s), \infty)$, both by definition. In both cases, $\mathfrak{P}_0^s$ is a polygon with the extremal points $(1, 0)$ and $(0, e(s))$, respectively, obtained by stationary policies.

For the induction step, fix n and a state s. We prove that $(p, E) \in \mathfrak{P}_{n+1}^s$ iff there exist a distribution $w : A(s) \to [0, 1]$ and achievable points $(p_{a,s'}, E_{a,s'}) \in \mathfrak{P}_n^{s'}$ such that

$$(p, E) = \sum_{a \in A(s), s' \in S} w(a) \cdot \Delta(s, a, s') \cdot (p_{a,s'}, E_{a,s'})$$

This equality follows from the interpretation of the Pareto set: For all actions a and successors s' there exists a policy $\pi_{a,s'}$ such that after n steps at least a fraction of $p_{a,s'}$ executions have reached the goal and the expected time to reach the goal is at most $E_{a,s'}$. So, we can take one step and then simply follow these respective policies to achieve the values in the equation. Dually, if there is a policy π for $(p, E) \in \mathfrak{P}_{n+1}^s$, we immediately get policies achieving the respective values in the successors (note the similarity to regular value iteration). The claim follows by the hypothesis. $\square$

This proof also yields an effective way of computing $\mathfrak{P}_n^s$.

Corollary 2. *We have that*

$$\mathfrak{P}_{n+1}^s = \mathrm{conv}\left(\bigcup_{a \in A(s)} \bigoplus_{s' \in S} \Delta(s, a, s') \cdot \mathfrak{P}_n^{s'}\right),$$

where $\oplus$ is the Minkowski sum and conv the convex hull.

With these results, we are ready to present our value iteration approach in Algorithm 1. As expected, it computes $\mathfrak{P}_n^s$ for increasing n using Corollary 2 and derives the optimal obtainable CVaR assuming that the VaR is at most n using Lemma 7. On top, the algorithm uses Lemma 4 as stopping criterion, ultimately yielding the optimal CVaR. Note that the algorithm can compute the optimal CVaR for several thresholds t simultaneously at essentially no additional cost: While computing the CVaR for the smallest threshold, the CVaR corresponding to the other thresholds can be obtained as an intermediate result.

Theorem 4. *Algorithm 1 is correct, i.e. always terminates and returns the optimal CVaR. Moreover, it is EXPTIME.*

Algorithm 1: Value Iteration to compute CVaR

Input: MDP $\mathcal{M}$, threshold t
Output: Optimal CVaR_t
  $c \leftarrow \infty, n \leftarrow 0$
  while $n \leq c$ **do**
    Compute $\mathfrak{P}_n^s$ for all $s \in S$
    $c_n \leftarrow n + \frac{1}{t} \cdot \left(\min\{E \mid (1-t, E) \in \mathfrak{P}_n^{\hat{s}}\} \cup \{\infty\} \right)$
    $c \leftarrow \min(c, c_n), n \leftarrow n + 1$
  return c

Total Cost

To extend our approach to the general scenario of total cost, only minor adjustments are necessary. We omit the completely analogous proofs of correctness. Note that both cases show that instead of Markovian policies, we now require policies which (only) depend on the total incurred cost.

Linear Programming   Recall that in order to obtain the LP approach we effectively "unrolled" the MDP, augmenting the state space with a step counter. We change this counter to track the accumulated cost, i.e. require

$$p_{s',i} = \sum_{s \in S, a \in A(s)} p_{s,a,i-C(s,a)} \cdot \Delta(s,a,s')$$

for $i < n$, adapting appropriately at the boundary. Informally, $p_{s,a,i}$ now corresponds to the probability of taking action a in state s when the total accumulated cost so far is i.

Around the VaR-guess n, additional care is needed: For example, suppose we are at state s with an accumulated cost of $n - 1$ and take action a with cost $C(s,a)$. In the LP, we would thus consider the variable $p_{s',n-1+C(s,a)}$. Now, for $C(s,a) > 1$, we need to modify the objective to also consider this part of the flow. In particular, we change the objective function to $\sum_{s \in S, 0 \leq c < C_{\max}} p_{s,n+c} \cdot (e(s) + c)$, where $C_{\max} = \max C(s,a)$ is the maximal cost.

Value Iteration   The Pareto set describes a trade-off between probability of reaching and expected remaining steps at n steps. We again change the interpretation of n to a cost bound. This means that a point (p, E) is in $\mathfrak{P}_n^s$ iff the goal can be reached with probability at least p while incurring a cost of at most n and at the same time have an expected cost of at most E to reach the goal afterwards. All statements can be replicated analogously, in particular we arrive at

$$\mathfrak{P}_n^s = \text{conv}\left(\bigcup_{a \in A(s)} \bigoplus_{s' \in S} \Delta(s,a,s') \cdot \mathfrak{P}_{n-C(s,a)}^{s'} \right),$$

where $\mathfrak{P}_n^s = \emptyset$ for $n < 0$. Consequently, we need to store the last $C_{\max}$ Pareto sets for each state in the VI algorithm.

Zero Cost States   We assumed that there are no zero-cost actions for simplicity. This assumption is implicitly applied in the above arguments. We can however "inline" such states through additional pre-computation: Intuitively, for a zero-cost action a, we can compute all possible ways the system could evolve after choosing a while restricted to using only zero-cost actions, and then replace the outcome of a with these options. The technical details of this procedure however are quite involved and beyond the scope of this work.
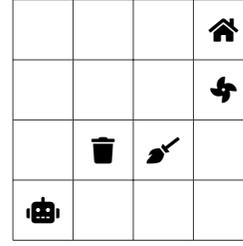

Figure 5: Visual representation of *Grid* for size 4x4. The robot currently is at $(1,1)$, while the janitor is at $(3,2)$.

Evaluation

We implemented prototypes of our algorithms in Java (Oracle JVM 17.0.1), delegating LP calls to `Gurobi` 9.1.2, running on consumer-grade hardware (AMD Ryzen 5 3600, 3.60 Ghz, 16 GB RAM). The JVM is limited to 10 GB of RAM through `-Xmx10G`. We augmented the VI approach by parallel computation and implemented a tailored Minkowski sum / convex hull computation. The LP approach computes a lower bound on the minimal VaR via a reachability VI. These optimizations already yield order-of-magnitude improvements compared to a naive implementation. Our implementation, all models, and instructions to reproduce the experiments can be found at https://doi.org/10.5281/zenodo.5764140.

Since (Chow et al. 2015) solve a similar problem (albeit with discounting), we tried evaluating their approach with a sufficiently large discounting factor, too. Unfortunately, we could not obtain an implementation of their methods.

Models   To test out our methods, we first consider a hand-crafted probabilistic path planning task on a grid world, called **Grid**. A robot has to move to its charging station, avoiding fixed obstacles. Moreover, a janitor is moving semi-randomly through a part of the region. In particular, the janitor faces a direction and either moves into that direction or randomly turns to one side. Whenever the robot is close to the janitor (distance ≤ 1), it is not allowed to move in order to avoid being stepped on. We consider the problem for several grid widths to investigate scalability. In order to keep the probability of interaction with the janitor high, we always restrict the janitor to a 4x4 grid that the robot necessarily has to cross. Furthermore, we consider two models from the literature, namely **FireWire** (Kwiatkowska, Norman, and Sproston 2003), the IEEE 1394 "FireWire" root contention protocol, and **WLAN** (Kwiatkowska, Norman, and Sproston 2002), the CSMA/CA mechanism of the 802.11 Wireless LAN protocol. See (Kwiatkowska et al. 2006) for further details on how **FireWire** and **WLAN** are constructed.

To evaluate the influence of the threshold, we consider another hand-crafted model **Walk**: Here, the system moves along a line of length n and can at each position choose to take one step forward, succeeding with 50% probability, or gamble to double its current position. The doubling action has a 10% chance to fail, instead halving the current position, and is disabled if it fails thrice. Note that the risk of the gamble, i.e. how much is "lost" in case of a fail, changes with the current position of the system. As such, depending on the

Model	$|S|$	$|A|$	$|\Delta|$	$\mathbb{E}$	$\mathrm{VaR}_{10\%}$	$\mathrm{CVaR}_{10\%}$	SSP	LP	VI
Grid($x = 4$)	1,270	4,230	12,390	6.7	8	11.0	0s	2s	0s
Grid($x = 8$)	3,196	12,796	37,996	13.1	16	17.6	1s	199s	2s
Grid($x = 16$)	6,396	28,796	85,996	19.3	22	23.6	2s	2,373s	11s
Grid($x = 32$)	12,796	60,796	181,996	35.9	39	40.8	15s	>1h	221s
FireWire	138,130	302,654	304,826	166.2	167	167.0	3s	*MO*	3s
WLAN	87,345	157,457	177,639	48.0	61	62.3	1s	*MO*	1s

Table 1: Summary of our experiments. For each model we list, from left to right, the number of states, actions, and transitions, the expected total cost until goal states are reached, i.e. the classical SSP value, the considered threshold, resulting VaR and CVar, and finally the times required to compute the SSP values, CVaR via LP, and CVaR via VI, respectively. *MO* denotes a memout. To ease presentation, we only considered a threshold of $\mathtt{t} = 10\%$ for this experiment.
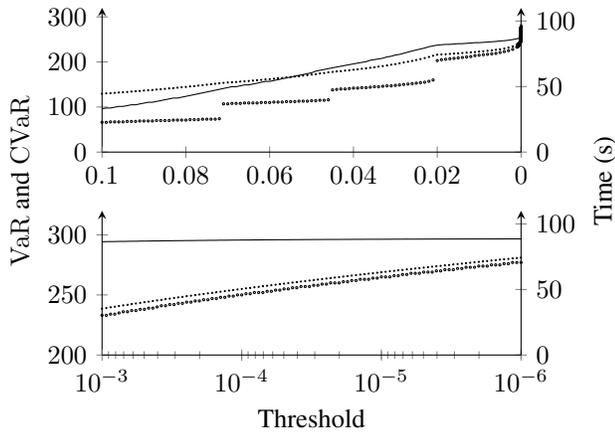

Figure 6: Runtime evaluation of our VI approach on *Walk* for different thresholds. We also depict the VaR and CVaR for each threshold by dots. Note that thresholds are *decreasing* from left to right. For readability, we include a zoomed, logarithmic plot for thresholds from 10^{-3} to 10^{-6}.

level of risk aversion, the system may choose the gambling option at different positions (or even not at all).

For simplicity, we consider uniform cost for all models.

Results Our results for the first experiment are summarized in Table 1. We clearly see that the LP approach quickly becomes infeasible, while VI can tackle significantly larger models. This is in line with the usual observations, where LP is more appealing in theory, but VI scales much better in practice. We highlight that the time required by VI is comparable to the time needed to simply optimize SSP. While solving the SSP first is required by our methods, it nevertheless is encouraging that the overhead of risk-aware optimization is not too large in these cases.

The results of the second experiment, evaluating the influence of the threshold, are depicted in Fig. 6. We omitted evaluating the LP approach here, since it took over 30 minutes to evaluate a single threshold. We clearly see the points where the optimal strategy switches away from taking the risky doubling action by a sharp increase of VaR. Moreover, the VaR and CVaR increase mostly linearly with the threshold, only spiking exponentially for very small thresholds.

This is to be expected due to Lemma 1: a small portion of probability mass remains inside the system for a long time. However, the time required for the VI steps decreased drastically, since the number of points in $\mathfrak{P}_n^s$ decreased. After this many steps, only a single dominant strategy remains, and most Pareto sets actually are singletons. Altogether, we observe that the runtime of VI seems to depend mostly linearly on the threshold, even on an adversarially crafted model.

Improvements As our simple optimization heuristics already yielded significant improvements, there likely are many further possibilities. We conjecture that additional structural properties might be used to speed up computation of $\mathfrak{P}_n^s$, e.g. a special structure of optimal policies. Moreover, we found that the performance of VI improves if we merge extremal points of $\mathfrak{P}_n^s$ which are, for example, very close to each other or lie just on boundary of the convex hull (i.e. removing them barely changes $\mathfrak{P}_n^s$). Since Bellman operators typically are contractive, we conjecture that the error introduced by this merging can be bounded, allowing for a trade-off between precision and speed. More generally, we think that in order to achieve a given precision of ε, polynomially in $\frac{1}{\varepsilon}$ and $\log \mathtt{t}$ many points for $\mathfrak{P}_n^s$ may be sufficient. Here, the ideas of (Papadimitriou and Yannakakis 2000) could be applicable.

Conclusion

We have presented a new risk-aware perspective on stochastic shortest path through the lens of CVaR. For this objective, we have derived an LP and a VI based solution, both of which yield precise, provably correct results. This analysis naturally comes at an additional price, however our experiments show that already with a simple implementation, our approach is feasible on moderately complex problems.

For future work, we aim to provide tight complexity bounds for **CVaR-SSP**. In (Bonet 2007), a rather general condition for polynomial convergence of VI for SSP is presented, which might be applicable to our approach, too. Moreover, we seek to study the exact structure of optimal policies. In particular, we conjecture that they do not alternate between actions. For the practical side, we plan to investigate the improvements mentioned in the previous section as well as study the influence of fixed-precision rounding errors. Finally, we want to investigate how risk-aware policies differ from purely expectation maximizing solutions in practice.

Acknowledgements

References

Artzner, P.; Delbaen, F.; Eber, J.-M.; and Heath, D. 1999. Coherent measures of risk. *Mathematical finance*, 9(3): 203–228.

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT Press. ISBN 978-0-262-02649-9.

Balaji, N.; Kiefer, S.; Novotný, P.; Pérez, G. A.; and Shirmohammadi, M. 2019. On the Complexity of Value Iteration. In Baier, C.; Chatzigiannakis, I.; Flocchini, P.; and Leonardi, S., eds., *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, 102:1–102:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Bäuerle, N.; and Ott, J. 2011. Markov Decision Processes with Average-Value-at-Risk criteria. *Math. Methods Oper. Res.*, 74(3): 361–379.

Beder, T. S. 1995. VAR: Seductive but dangerous. *Financial Analysts Journal*, 51(5): 12–24.

Bellman, R. 1966. Dynamic programming. *Science*, 153(3731): 34–37.

Bertsekas, D. P. 2005. *Dynamic programming and optimal control, 3rd Edition*. Athena Scientific. ISBN 1886529264.

Bertsekas, D. P.; and Tsitsiklis, J. N. 1991. An Analysis of Stochastic Shortest Path Problems. *Math. Oper. Res.*, 16(3): 580–595.

Bertsekas, D. P.; and Tsitsiklis, J. N. 1996. *Neuro-dynamic programming*, volume 3 of *Optimization and neural computation series*. Athena Scientific. ISBN 1886529108.

Bonet, B. 2007. On the Speed of Convergence of Value Iteration on Stochastic Shortest-Path Problems. *Math. Oper. Res.*, 32(2): 365–373.

Borkar, V. S.; and Jain, R. 2014. Risk-Constrained Markov Decision Processes. *IEEE Trans. Autom. Control.*, 59(9): 2574–2579.

Carpin, S.; Chow, Y.; and Pavone, M. 2016. Risk aversion in finite Markov Decision Processes using total cost criteria and average value at risk. In *ICRA*, 335–342. IEEE.

Chatterjee, K.; Kretínská, Z.; and Kretínský, J. 2017. Unifying Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes. *Log. Methods Comput. Sci.*, 13(2).

Chow, Y.; Tamar, A.; Mannor, S.; and Pavone, M. 2015. Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 1522–1530.

Filippi, C.; Guastaroba, G.; and Speranza, M. G. 2020. Conditional value-at-risk beyond finance: a survey. *Int. Trans. Oper. Res.*, 27(3): 1277–1319.

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. *Comb.*, 4(4): 373–396.

Keramati, R.; Dann, C.; Tamkin, A.; and Brunskill, E. 2020. Being Optimistic to Be Conservative: Quickly Learning a CVaR Policy. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 4436–4443. AAAI Press.

Khachiyan, L. G. 1979. A polynomial algorithm in linear programming. In *Doklady Academii Nauk SSSR*, volume 244, 1093–1096.

Křetínský, J.; and Meggendorfer, T. 2018. Conditional Value-at-Risk for Reachability and Mean Payoff in Markov Decision Processes. In Dawar, A.; and Grädel, E., eds., *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, 609–618. ACM.

Kwiatkowska, M. Z.; Norman, G.; Parker, D.; and Sproston, J. 2006. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods Syst. Des.*, 29(1): 33–78.

Kwiatkowska, M. Z.; Norman, G.; and Sproston, J. 2002. Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol. In Hermanns, H.; and Segala, R., eds., *Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Second Joint International Workshop PAPM-PROBMIV 2002, Copenhagen, Denmark, July 25-26, 2002, Proceedings*, volume 2399 of *Lecture Notes in Computer Science*, 169–187. Springer.

Kwiatkowska, M. Z.; Norman, G.; and Sproston, J. 2003. Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol. *Formal Aspects Comput.*, 14(3): 295–318.

Meggendorfer, T. 2022. Risk-aware Stochastic Shortest Path. *CoRR*, abs/2203.01640.

Papadimitriou, C. H.; and Tsitsiklis, J. N. 1987. The Complexity of Markov Decision Processes. *Math. Oper. Res.*, 12(3): 441–450.

Papadimitriou, C. H.; and Yannakakis, M. 2000. On the Approximability of Trade-offs and Optimal Access of Web Sources. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, 86–92. IEEE Computer Society.

Piribauer, J. 2021. *On Non-Classical Stochastic Shortest Path Problems*. Ph.D. thesis, Technische Universität Dresden.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley. ISBN 978-0-47161977-2.

Rockafellar, R. T.; Uryasev, S.; et al. 2000. Optimization of conditional value-at-risk. *Journal of risk*, 2: 21–42.

Ruszczynski, A. 2010. Risk-averse dynamic programming for Markov decision processes. *Math. Program.*, 125(2): 235–261.

Sarykalin, S.; Serraino, G.; and Uryasev, S. 2008. Value-at-risk vs. conditional value-at-risk in risk management and optimization. In *State-of-the-art decision-making tools in the information-intensive age*, 270–294. Informs.

Schrijver, A. 1999. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley. ISBN 978-0-471-98232-6.

Shapiro, A.; Dentcheva, D.; and Ruszczynski, A. 2014. *Lectures on Stochastic Programming - Modeling and Theory, Second Edition*, volume 16 of *MOS-SIAM Series on Optimization*. SIAM. ISBN 978-1-61197-342-6.

Tamar, A.; Chow, Y.; Ghavamzadeh, M.; and Mannor, S. 2017. Sequential Decision Making With Coherent Risk. *IEEE Trans. Autom. Control.*, 62(7): 3323–3338.

White, D. J. 1985. Real applications of Markov decision processes. *Interfaces*, 15(6): 73–83.

White, D. J. 1993. A survey of applications of Markov decision processes. *Journal of the operational research society*, 44(11): 1073–1096.