Reinforcement Learning of Risk-Constrained Policies in Markov Decision Processes (Extended Abstract)

Tomáš Brázdil[1], **Krishnendu Chatterjee**[2], **Petr Novotný**[1], **Jiří Vahala**[1]
[1]Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xbrazdil, petr.novotny, xvahala1}@fi.muni.cz
[2]Institute of Science and Technology Austria, Klosterneuburg, Austria
Krishnendu.Chatterjee@ist.ac.at

Abstract

Markov decision processes (MDPs) are the standard model of sequential decision making under stochastic uncertainty. A classical optimization criterion for MDPs is to maximize the expected discounted-sum payoff, which ignores low probability catastrophic events with highly negative impact on the system. On the other hand, risk-averse policies require the probability of undesirable events to be below a given threshold, but they do not account for optimization of the expected payoff. We consider MDPs with discounted-sum payoff and with failure states which represent catastrophic outcomes. The objective of *risk-constrained* planning is to maximize the expected discounted-sum payoff among risk-averse policies that ensure the probability to encounter a failure state is below a desired threshold. Our main contribution is an efficient risk-constrained planning algorithm that combines UCT-like search with a predictor learned through interaction with the MDP (in the style of AlphaZero) and with a risk-constrained action selection via linear programming. We demonstrate the effectiveness of our approach with experiments on classical MDPs from the literature, including benchmarks with an order of 10^6 states.

This extended abstract summarizes results presented in the paper *Reinforcement Learning of Risk-Constrained Policies in Markov Decision Processes* published at AAAI'20.

1 Introduction & Problem Statement

MDPs. We consider *Markov decision processes* (MDPs) with discounted-sum payoff, a standard model of probabilistic decision-making (Puterman 1994). Formally and MDP consists of: a finite set $\mathcal{S}$ of *states;* a finite alphabet $\mathcal{A}$ of *actions;* a *probabilistic transition function* δ that given a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$ returns the probability distribution $\delta(s,a)$ over the successor states; a *reward function* $rew : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; and a *discount factor* γ.

Fixing some initial state s_0, the interaction with an MDP starts in s_0 and proceeds sequentially through a *policy* π, a function which acts as a blueprint for selecting actions, producing longer and longer *history* of actions and observations. We denote by $\mathbb{P}^\pi(E)$ the probability of an event E under policy π, and by $\mathbb{E}^\pi[X]$ the expected value of a random variable X under π.

Expectation optimization and risk. In the classical studies of MDPs with discounted-sum payoff, the objective is to obtain policies that maximize the *expected* payoff. Formally, the expected payoff of a policy π is the number $Payoff(\pi) = \mathbb{E}^\pi[\sum_{i=0}^{\infty} \gamma^i \cdot rew(s_i, a_i)]$, where s_i, a_i are the current state and the action selected in step i. However, the expected payoff criterion ignores the possible presence of low probability failure events that can have a highly negative impact.

Motivating scenarios. In scenarios such as a robot exploring an unknown environment, a significant damage of the robot ends the mission. Of course, the policy which never moves the robot would be likely safe in such a scenario, but goes against the robot's primary objective of an effective exploration. Instead, the operator can set a *risk threshold,* i.e. the probability of the robot's destruction that is acceptable under given operational parameters. The goal is to ensure effective exploration while keeping the risk of destruction below the threshold, which naturally gives rise to the risk-constrained planning problem we consider.

In the pure expectation-optimizing framework, we might attempt to encode the risk-taking aspect directly into the reward function, e.g. by stipulating that entering a failure state incurs a large negative penalty. However, the risk-taking aspect of the resulting policy is then very sensitive to the penalty variations, demanding an elaborate tuning of the penalties to achieve a desired behaviour (see also (Undurti and How 2010) for a critical discussion of reward-function engineering in risk-constrained scenarios). In contrast, we aim to achieve an explicit control over the risk taken by a policy, decoupling the risk-taking aspect from the expected payoff optimization.

Problem statement. Given an MDP, we fix a set $F \subseteq \mathcal{S}$ of *failure states*. A *risk* of a policy π is then the probability that a failure state is encountered:

$$Risk(\pi) = \mathbb{P}^\pi\Big(\bigcup_{i=0}^{\infty} \{s_i \in F\}\Big).$$

We assume that each $s \in F$ is a *sink,* i.e. $\delta(s,a)(s) = 1$ and $rew(s,a) = 0$ for all $a \in \mathcal{A}$. Hence, F models failures after which the agent has to cease interacting with the environment (e.g. due to being destroyed).

The risk-constrained planning problem is defined as follows: given an MDP $\mathcal{M}$ and a *risk threshold* $\Delta \in [0,1]$, find

a policy π which maximizes $Payoff(\pi)$ subject to the constraint that $Risk(\pi) \leq \Delta$. If there is no *feasible* policy, i.e. a policy s.t. $Risk(\pi) \leq \Delta$, then we want to find a policy that minimizes the risk and among all such policies optimizes the expected payoff.

The risk-constrained planning problem can be formulated as a special case of constrained MDPs (Altman 1999) (see (Brázdil et al. 2020) for an overview of related work). Our main contribution is a new reinforcement-learning algorithm for risk-constrained planning.

2 Our Contribution

We present RAlph (a portmanteau of "Risk" and "Alpha"), an online algorithm for risk-constrained planning. Inspired by the successful approach of AlphaZero (Silver et al. 2017; 2018), RAlph combines a UCT-like exploration of the *history tree* $\mathcal{T}$ of the MDP with evaluation of the leaf nodes via a suitable *predictor* learned through a repeated interaction with the system. On top of this, we augment the algorithm's action-selection phase with a risk-constrained mechanism based on evaluation of a linear program over the constructed tree. The algorithm starts with a risk threshold $\Delta_0 = \Delta$, which is updated in each decision step to take into account the risk already taken by the agent. We denote by Δ_i the threshold in step i.

The main novel features of RAlph (in comparison with the AlphaZero framework) are the following:

Risk predictor. In our algorithm, the predictor is extended with risk prediction.

Risk-constrained action selection. When selecting an action a_i to be played at step i, we solve a linear program (LP) over $\mathcal{T}$, yielding a local policy that maximizes the estimated payoff while keeping the estimated risk below the current threshold Δ_t. The distribution ξ_i used by the local policy in the first step is then used to sample a_i. The LP is such that if the predictor was replaced with a perfect oracle, the LP solution would give the optimal risk-constrained policy.

Risk-constrained exploration. Some variants of AlphaZero enable additional exploration by selecting each action with a probability proportional to its exponentiated visit count (Silver et al. 2017). Our algorithm uses a technique which perturbs the distribution computed by the LP while keeping the risk estimate of the perturbed distribution below the required threshold

Estimation of alternative risk. The risk threshold must be updated after playing an action, since each possible outcome of the action has a potential contribution towards the global risk. We use linear programming and the risk predictor to obtain an estimate of these contributions.

3 Implementation & Evaluation

Predictor. In principle any predictor (e.g. a neural net) can be used with RAlph. In our implementation, as a proof of concept, we use just a simple table predictor, directly storing the estimated payoff and risk for each state s.

Benchmarks. We implemented RAlph and evaluated it on two types of benchmarks. The first is a perfectly observable version of Hallway (Pineau et al. 2003; Smith and Simmons 2004) where we control a robot in a grid maze. In each move, there is a chance of the robot being shifted sideways of the intended move direction, possibly into a trap where there is a chance of destruction. As a second type, we consider a controllable 1-dimensional *random walk,* modeling an investor in a financial market.

```
1 1 1 1 1 1
1 A B x   1
1 D C E g 1
1 1 1 1 1 1
```

Figure 1: Example of a Hallway MDP. Symbols '1', 'x', 'g' represent wall/trap/goal cell respectively. The agent starts in B facing east, and obtains a reward for reaching the goal cell.

Evaluation. We evaluate RAlph on four instances of the Hallway benchmark. The corresponding MDPs have state-spaces of sizes equal to 20, 44, 1136, and 6553600, respectively. For the random walk, we consider benchmarks with 50 and 200 states. We compared RAlph with the RAMCP algorithm from (Chatterjee et al. 2018). RAMCP can also perform risk-averse planning via the use of heuristic search and linear programming, but does not use any predictor. The outcome of our experiments is reported in (Brázdil et al. 2020). RAlph was shown to be much faster than RAMCP, making up to two orders of magnitude less node expansions when building $\mathcal{T}$. RAlph also consistently finds solutions that satisfy the risk threshold, which is not always the case for RAMCP, whose expected payoff and risk tended to deteriorate quite fast with increasing number of states. In contrast, RAlph was able to learn a well-behaving risk-averse policy in less than 15 minutes even in the largest benchmark.

Discussion. RAlph exhibited interesting behavior on several benchmarks. An an example, consider the Hallway instance shown in Figure 1. For $\Delta = 0$, the only way to reach the gold is by exploiting the move perturbations: since the robot cannot to move east from C without risking a shift to the trap, it must keep circling through A, B, C, D until it is randomly shifted to E. RAlph is able, with some parameter tuning, to find this policy.

4 Conclusions & Future Work

Our experiments show that even with a simple predictor, RAlph performs and scales significantly better than a state-of-the-art algorithm. As an interesting future work we see extension of the method to POMDPs and incorporation of more sophisticated predictors.

Acknowledgements

Krishnendu Chatterjee is supported by the Austrian Science Fund (FWF) NFN Grant No. S11407-N23 (RiSE/SHiNE), and COST Action GAMENET. Tomáš Brázdil is supported by the Grant Agency of Masaryk University grant no. MUNI/G/0739/2017 and by the Czech Science Foundation grant No. 18-11193S. Petr Novotný and Jiří Vahala are supported by the Czech Science Foundation grant No. GJ19-15134Y.

References

Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

Brázdil, T.; Chatterjee, K.; Novotný, P.; and Vahala, J. 2020. Reinforcement learning of risk-constrained policies in Markov decision processes. *CoRR* abs/2002.12086.

Chatterjee, K.; Elgyütt, A.; Novotný, P.; and Rouillé, O. 2018. Expectation optimization with probabilistic guarantees in pomdps with discounted-sum objectives. In *IJCAI 2018*, 4692–4699.

Pineau, J.; Gordon, G.; Thrun, S.; et al. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, 1025–1032.

Puterman, M. L. 1994. *Markov Decision Processes*. J. Wiley and Sons.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362(6419):1140–1144.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *UAI*, 520–527. AUAI Press.

Undurti, A., and How, J. P. 2010. An online algorithm for constrained POMDPs. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 3966–3973. IEEE.