Robust Online Planning with Imperfect Models

Maxim Rostov & Michael Kaisers

23th of April, 2022

ALA 2021, 3 & 4 May 2021, London (Virtual)

Table of contents

Motivation & Challenges

Motivation: control of real-life safety and business critical systems

(a)

(b)

Challenges:

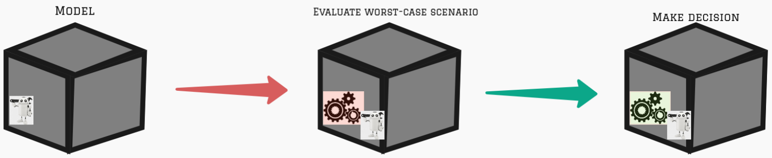
- stochastic factors and adversaries may introduce uncertainty to the system
- real-world system are often large and complex

(a) https://www.solvay.com/en/article/rise-of-stationary-battery

(b) https://www.smmt.co.uk/2018/11/feature-how-scheduling-software-can-assist-future-electric-buses/

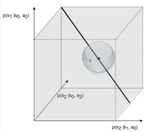
Approach

Enter **sequential decision-making** where only an **approximate** (sample) model of the environment is given

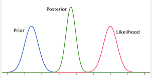


- The whole **system's description** (model) is used to find decisions
- Worst case over uncertainties induces **robustness**
- Only part of the systems is considered at a time: online **decision-time** planning

Online Robust Approach



Deterministically defined error bounds and performance guarantees



No prior distribution over model errors is assumed

Online planning provides for better efficiency within large/complex systems

Table of contents

Markov Decision Process

Markov Decision Process (MDP) framework [5] . MDP model $\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ is a (discrete) state space, $\mathcal{A}$ is a (discrete) action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $\gamma \in [0, 1)$

- $\mathcal{M}_\phi \equiv \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\phi, \mathcal{R}, \gamma \rangle$
- $\pi \equiv \pi_\theta$
- Trajectory density function ρ is used to generate trajectories

$$\rho_{\phi,\theta}(\tau) = \mu_0 \left(s_0\right) \pi_\theta \left(a_0 | s_0\right) \prod_{t=1}^{T-1} \mathcal{P}_\phi \left(s_{t+1} | s_t, a_t\right) \pi_\theta \left(a_t | s_t\right) \tag{1}$$

  where μ is the stationary initial state distribution.

- Policy is **evaluated** with:

$$G(\tau) = \sum_{i=0}^{T-1} \gamma^t \mathcal{R} \left(s_t, a_t\right) \tag{2}$$

$$Q^\pi (s, a) = \mathbb{E}_{\tau \sim \rho_{\phi,\theta}} [G(\tau) \mid s_0 = s,\ a_0 = a] \tag{3}$$

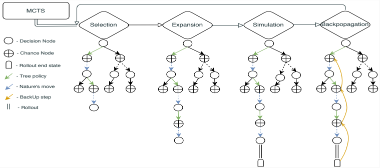
$$V^\pi (s) = \mathbb{E}_{\tau \sim \rho_{\phi,\theta}} [G(\tau) \mid s_0 = s] \tag{4}$$

Solving MDP's

Background planning: Value Iteration is an iterative model-based method that calculates the expected utility of each state using the values of the all other states in $\mathcal{S}$

$$V(s) \leftarrow \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{\phi}\left(s'|s,a\right) V(s') \right] \qquad \forall s \in \mathcal{S}. \qquad (5)$$

Decision-time planning: Monte Carlo Tree Search grows a tree of possible path continuations for a root state. Our MCTS use a UCT [1] heuristic evaluation function to decide on the direction of the tree search.



Non-robust policy evaluation

$$f(\pi) = \mathbb{E}_{\tau \sim \rho_{\phi,\theta}} [G(\tau)] \tag{6}$$

Robust policy evaluates the worst-case expected return (WCER) [6]:

$$f_{\mathcal{U}}(\pi) = \min_{\phi \in \mathcal{U}_\phi} \mathbb{E}_{\tau \sim \rho_{\phi,\theta}} [G(\tau)] \tag{7}$$

Where

$$\mathcal{U}_\phi = \underset{(s,a)\in\mathcal{S}\times\mathcal{A}}{\times} \mathcal{U}_\phi(p_0(\cdot|s,a), \epsilon) \tag{8}$$

and $p_0(\cdot|s,a) \subseteq \mathbb{R}_+^{|\mathcal{S}|}$ is a marginal pmf from $\mathcal{P}_0$ and $p(\cdot|s,a) \in \mathcal{U}(p_0(\cdot|s,a), \epsilon)$ where $d\left(p(\cdot|s,a), p_0(\cdot|s,a)\right) \leq \epsilon$.

Robust Value Iteration

Robust evaluation is applied to the value function:

$$V^{\pi}_{\phi_{min}}(s) = \min_{\phi \in \mathcal{U}_\phi} \mathbb{E}_{\tau \sim \rho_{\phi,\theta}} \left[G(\tau) \mid s_0 = s \right] \qquad \forall s \in \mathcal{S}. \tag{9}$$

Use **two-stage** robust optimization procedure to find the optimal robust policy:

$$Q^{\tilde{\pi}}(s,a) = \min_{\phi \in \mathcal{U}_\phi} \left[\mathcal{R}(s,a) + \gamma V^{\pi}_{\phi}(s') \right] \qquad \forall s, s' \in \mathcal{S};\ \forall a \in \mathcal{A}, \tag{10}$$

$$V^{\tilde{\pi}}(s) = \max_{a \in \mathcal{A}} Q^{\tilde{\pi}}(s,a) \qquad \forall s \in \mathcal{S};\ \forall a \in \mathcal{A}. \tag{11}$$

Therefore, the update formula for **robust VI** (rVI) [2, 4]

$$V(s) \leftarrow \max_{a \in \mathcal{A}} \left[\min_{\phi \in \mathcal{U}_\phi} \left(\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_\phi(s'|s,a) V(s') \right) \right] \qquad \forall s \in \mathcal{S}. \tag{12}$$

Table of contents

Solving Minimization Stage

Minimization program:

Project the reference model parameters of $p_0(\cdot|s,a)$ onto the allowed space $\mathcal{U}(p_0(\cdot|s,a),\epsilon)$.

$$\begin{aligned} \underset{p(\cdot|s,a)}{\text{minimize}} \quad & \sum_{s' \in \mathcal{S}} p(s'|s,a) V^{\pi}(s'), \\ \text{subject to} \quad & \sum_{s' \in \mathcal{S}} p(s'|s,a) = 1, \\ & p(s'|s,a) \in [0,1], \qquad \forall s' \in \mathcal{S} \\ & d\left(p(\cdot|s,a), p_0(\cdot|s,a)\right) \leq \epsilon \end{aligned} \tag{13}$$

Solve with:

- **Direct Projection**: find the worst-case model $\mathcal{P}_{min}$ by directly moving towards the boundary of $\mathcal{U}$. -> **Algorithm 1**
- **Indirect Projection**: move to the worst-case model by making small steps towards it, i.e., akin to gradient descent. -> **Algorithm 2**

Critical Assumptions on Robustness: Locality

Approximate DP solution via **decision-time planning**, in our case with Monte Carlo Tree Search.

Local robustness: $V^\pi(s')$ is not known for all s', i.e., only an estimate $\hat{V}^\pi(s')$ on the state sub-space $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ is available.

$$\begin{aligned} \underset{p(\cdot|s,a)}{\text{minimize}} \quad & \sum_{s' \in \tilde{\mathcal{S}}} p(s'|s,a)\hat{V}^\pi(s'), \\ \text{subject to} \quad & \sum_{s' \in \tilde{\mathcal{S}}} p(s'|s,a) = 1, \\ & p(s'|s,a) \in [0,1], \qquad \forall s' \in \tilde{\mathcal{S}} \\ & d\left(p(\cdot|s,a), p_0(\cdot|s,a)\right) \leq \epsilon \end{aligned} \tag{14}$$

Robust MCTS with batch updates

rMCTS-Batched: use batches of rollouts to update the estimation of $\mathcal{P}_{min}$ after each batch based on the inferred (robust) value function

Algorithm 3: rMCTS-Batched

Input : $\mathcal{M}_0$, reference MDP $\mathcal{M}_0 = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_0, \mathcal{R}, \gamma \rangle$
s_{root}, root state
$n_{batches}$, number of batches
n_{sims}, number of simulations for a batch
ϵ, error bound

Output: a, robust action $\arg\max_a Q(s_{root}, a)$

Init: $\mathrm{Q}_{\mathrm{r}}, \mathrm{N}_{\mathrm{r}}$, empty hash tables

$\hat{\mathcal{P}}_{min} \xleftarrow{copy} \mathcal{P}_0$

for i **in** $[1, \ldots, n_{batches}]$ **do**

  $\mathcal{M}_{min} = \langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}_{min}, \mathcal{R}, \gamma \rangle$

  $\mathrm{Q}_{\mathrm{r}}, \mathrm{N}_{\mathrm{r}} \leftarrow \mathrm{MCTS}\big(\mathrm{s}_{\mathrm{root}}, \mathcal{M}_{min}, \mathrm{n}_{\mathrm{sims}}, \mathrm{Q}_{\mathrm{r}}, \mathrm{N}_{\mathrm{r}}\big)$

  $\mathrm{V}_{\mathrm{r}} \leftarrow \mathrm{TakeArgMax}(\mathrm{Q}_{\mathrm{r}})$

  $\tilde{\mathcal{S}} \times \tilde{\mathcal{A}} \leftarrow \mathrm{GetStateActionTuples}(\mathrm{Q}_{\mathrm{r}})$

  $\hat{\mathcal{P}}_{min} \leftarrow \text{Algorithm 1}\Big(\mathcal{P}_0, \mathrm{V}_{\mathrm{r}}, \tilde{\mathcal{S}} \times \tilde{\mathcal{A}}, \epsilon\Big)$

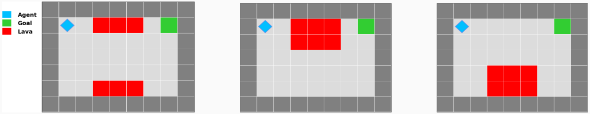
end

rMCTS-Iterative: batches of size 1 and ”small” minimization steps with indirect projection operator (Algorithm 2).

Table of contents

LavaWorld

- LavaWold, or Distributional Shift environment as presented in DeepMind's Satefy Gridworlds kit [3]
- Model spaces via Wasserstein distance

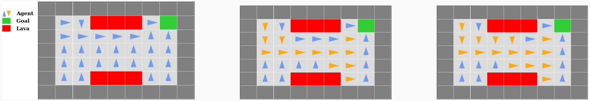


(c) (d) (e)

Figure 1: (a) Environment set-up which is given to an agent at train time. The agent is tested on environments (b) and (c).

LavaWorld: Evaluation

Differences between the policies derived with VI/MCTS vs rVI/rMCTS:



(a) Non-robust **(b)** Robust, $\epsilon = 0.2$ **(c)** Robust, $\epsilon = 0.4$

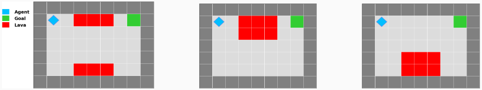
Figure 2: Policies derived via non-robust and robust (Wasserstein distance) methods

LavaWorld: Evaluation Cont.

Performance under model perturbations:

Table 1: Average discounted return over 10000 runs for s_0. The robust polices were derived with $\epsilon = 0.2$

	$\mathcal{P}_0$	$\mathcal{P}^{rVI}_{min}$	$\mathcal{P}^{rMCTSb}_{min}$	$\mathcal{P}^{rMCTSi}_{min}$	$\mathcal{P}_{test}$	$\mathcal{P}_0 + \xi$
π^{MCTS}	**0.698**	0.212	0.223	0.209	0.340	0.575 (0.239)
π^{rVI}	0.630	**0.309**	**0.302**	**0.304**	**0.625**	0.573 (0.123)
π^{rMCTSb}	0.626	**0.308**	**0.307**	**0.308**	**0.638**	0.574 (0.123)
π^{rMCTSi}	0.626	**0.302**	**0.308**	**0.302**	**0.631**	0.574 (0.120)



(a) (b) (c)

Conclusion

Main takeaways:

- **Modular projection operators** that can be integrated in a planning algorithm
- A novel family of **robust decision-time planning** techniques based on Monte Carlo Tree Search (rMCTS)
- Evidence into the value of robust optimization under model perturbations, i.e., **imperfect models**

Future work:

- Experiments with **larger state/action spaces**
- Consider **other error bounds** and corresponding projection operators

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 05 2002.

[2] G. Iyengar. Robust dynamic programming. *Math. Oper. Res.*, 30:257–280, 05 2005.

[3] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg. AI safety gridworlds. *CoRR*, abs/1711.09883, 2017.

[4] A. Nilim and L. Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53:780–798, 10 2005.

[5] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994.

[6] W. Wiesemann, D. Kuhn, and B. Rustem. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.