Approximate Robust Control of Uncertain Dynamical Systems

Anonymous Author(s)
Affiliation
Address
email

Abstract

This work studies the design of safe control policies for large-scale non-linear systems operating in uncertain environments. In such a case, the robust control framework is a principled approach to safety that aims to maximize the worst-case performance of a system. However, the resulting optimization problem is generally intractable for non-linear systems with continuous states. To overcome this issue, we introduce two tractable methods that are based either on sampling or on a conservative approximation of the robust objective. The proposed approaches are applied to the problem of autonomous driving.

1 Introduction

Reinforcement Learning is a general framework that allows the optimal control of a Markov Decision Process $(\mathcal{S}, \mathcal{A}, T, r)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r \in [0,1]^{\mathcal{S}\times\mathcal{A}}$ and unknown transition dynamics $T(s'|s,a) \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ by searching for the policy $\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}$ with maximal expected value v_π^T of the total discounted reward R_π^T:

$$R_\pi^T(s) \stackrel{\text{def}}{=} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t), \qquad v_\pi^T(s) \stackrel{\text{def}}{=} \mathbb{E}(R_\pi^T(s)), \tag{1}$$

where $s_0 = s$, $a_t \sim \pi(s_t)$, $s_{t+1} \sim T(s_{t+1}|s_t, a_t)$, $\gamma \in [0,1)$ is the discount factor and $\mathcal{M}(X)$ denotes the set of probability measures over X.

Unfortunately, its application to real-world tasks has so far been limited by its considerable need for experiences. It is generally recognized (Sutton, 1990; Atkeson and Santamaria, 1997) that the most sample-efficient approach is the family of model-based methods which learn a nominal model $\hat{T}$ of the environment dynamics that is leveraged for policy search:

$$\max_\pi v_\pi^{\hat{T}} \tag{2}$$

One drawback of such methods is that they suffer from model bias; that is, they ignore the error between the learned dynamics $\hat{T}$ and the real environment T. It has been shown that model bias can dramatically degrade the policy performances (Schneider, 1997).

Model errors can instead be explicitly considered and expressed through an *ambiguity set* of all possible dynamics models. Such a set can be constructed from a history of observations by computing the confidence regions associated with the system identification process (Iyengar, 2005; Nilim and El Ghaoui, 2005; Dean et al., 2017; Maillard, 2017). In this work, we will consider ambiguity sets of parametrized deterministic dynamical systems $s' = T_\theta(s,a)$ whose unknown parameters θ lie in a compact set Θ of $\mathbb{R}^p$.

In the optimal control framework, model uncertainty is handled by maximizing the *expected* performances with respect to unknown dynamics. In stark contrast, in real-world applications where failures may turn out very costly, the decision maker often prefers to minimize the risk of the policy, which

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

can be defined with several metrics characterizing the distribution of the policy outcome (García and Fernández, 2015).

The robust control framework is a popular setting in which the risk of a policy is defined as the worst possible outcome realization among the ambiguity set, to guarantee a lower-bound performance of the robust policy when executed on the true model:

$$\max_{\pi} \min_{T} v_{\pi}^{T} \tag{3}$$

Robust optimization has been studied in the context of finite Markov Decision Processes (MDP) with uncertain parameters by Iyengar (2005), Nilim and El Ghaoui (2005) and Wiesemann et al. (2013). They show that the main results of Dynamic Programming can be extended to their robust counterparts only when the dynamics ambiguity set verifies certain rectangularity properties. In the control theory community, the robust control problem is mainly restricted to the context of linear dynamical systems with bounded uncertainty in the time or frequency domain, where the objective is to guarantee stability (e.g. $\mathcal{H}_\infty$-optimal control, see Basar and Bernhard, 1996) or performance (e.g. LQ optimal control theory, see Petersen and Tempo, 2014). The existing nonlinear robust control approaches such as sliding mode control (Li et al., 2017), feedback linearization, backstepping, passivation and input-to-states stabilization (Khalil, 2014) are usually based on canonical representations of regulated dynamics and admit constructive numeric realizations for systems of rather low dimensions.

There have been few attempts of robust control of large-scale systems with both continuous states and non-linear dynamics, which is the focus of this paper. Our contribution is twofold. In section 2, we first consider a simpler case where the ambiguity set Θ and action space $\mathcal{A}$ are both finite and introduce a sampling-based planner that approximately maximizes the robust objective (3). In section 3, we move to continuous ambiguity sets and form a conservative relaxation of the robust policy evaluation problem using interval predictors. In section 4, we illustrate the benefits of both techniques (for discrete, versus continuous Θ) on a problem of tactical decision-making for autonomous driving.

2 Sampling-based planning

If the true dynamics model T_θ were known and the action-space $\mathcal{A}$ finite, sampling-based algorithms could be used to perform approximate optimal planning. In order to generalize to the robust setting, we need to make the following assumption about the structure of the ambiguity set:

Assumption 1 (Structure)**.** *The ambiguity set Θ and the action space $\mathcal{A}$ are discrete and finite:*

$$\mathcal{A} = \{a_k\}_{k\in[1,K]} \quad \textit{and} \quad \Theta = \{\theta_m\}_{m\in[1,M]} \tag{4}$$

We slightly abuse notation and denote $T_m = T_{\theta_m}$.

Such a structure of the ambiguity set typically stems directly from expert knowledge of the problem at hand. In general, it is nonrectangular, which implies that the Robust Bellman Equation does not hold (Wiesemann et al., 2013). This prevents us from building on planners that implicitly use this property and generate trajectories step-by-step by picking promising successor states, such as MCTS (Coulom, 2006) or UCT (Kocsis and Szepesvári, 2006). Instead, we turn to algorithms that perform optimistic sampling of entire sequences of actions and work directly at the leaves of the expanded tree (see, e.g. Bubeck and Munos, 2010). More precisely, we build on the work of Hren and Munos (2008) on optimistic planning for deterministic dynamics, which we extend to the robust setting.

We use similar notations and consider the infinite look-ahead tree $\mathcal{T}$ composed of all reachable states. Each node corresponds to a joint state $\{s_{m,t}\}_{m\in[1,M]}$ associated with the different dynamics T_m. The root starts at the current state, and all nodes have K children, each corresponding to an action $a_k \in \mathcal{A}$ and associated with the successor joint state $\{s_{m,t+1} = T_m(s_{m,t}, a_k)\}_{m\in[1,M]}$. We use the standard notations over alphabets to refer to nodes in $\mathcal{T}$ as action sequences. Thus, a finite word $i \in \mathcal{A}^*$ of length d represents the node obtained following the action sequence $(i_0, \cdots, i_d)$ from the root. Sequences $i \in \mathcal{A}^*$ and $j \in \mathcal{A}^*$ can be concatenated as $ij \in \mathcal{A}^*$, the set of suffixes of i is $i\mathcal{A}^\infty = \{j \in \mathcal{A}^\infty : \exists h \in \mathcal{A}^\infty$ such that $j = ih\}$, and the empty sequence is $\emptyset$.

The sample complexity is expressed in terms of number n of expanded nodes. It is related to the number of calls to dynamics models: when a node i is expanded, all successor states are computed for all K actions and M dynamics. At an iteration n, we denote $\mathcal{T}_n$ the tree of already expanded nodes, and $\mathcal{L}_n$ the set of its leaves.

Definition Fix a dynamics model $m \in [1, M]$. Hren and Munos (2008) define for any node $i \in \mathcal{T}$ of depth d the optimal value v_i^m, its lower bound u-value u_i^m and upper-bound b-value b_i^m. These variables depend on the dynamics m and will therefore be referred to with a superscript m notation.

We extend these dynamics-dependent variables to the robust setting, using superscript r in notations.

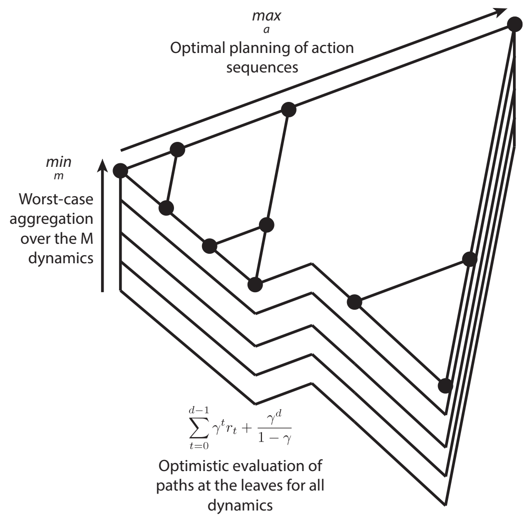


Figure 1: The computation of robust b-values in Algorithm 1. The simulation of trajectories for every dynamics model T_m is represented as stacked versions of the expanded tree $\mathcal{T}_n$.

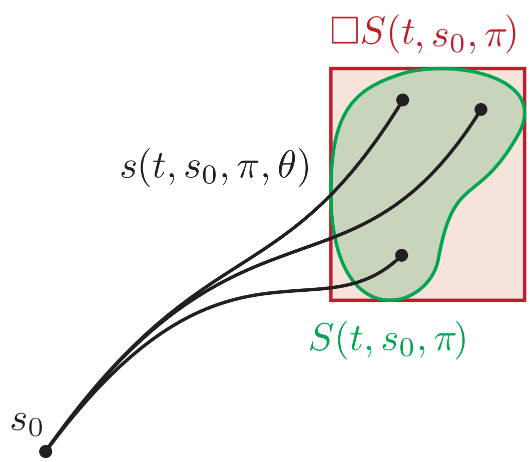


Figure 2: A few trajectories are sampled from an initial state s_0 following a policy π with various dynamics parameters θ_m (in black). The union of reachability sets is shown in green, and its interval hull in red.

- The robust value v_i^r of a path $i \in \mathcal{A}^*$ as the restriction of (3) to policies that start with the action sequence i:

$$v_i^r \stackrel{\text{def}}{=} \max_{\pi \in i\mathcal{A}^\infty} \min_{m \in [1,M]} R_\pi^{T_m} \tag{5}$$

  By definition, the robust value of (3) is recovered at the root $v_\emptyset^r = v^r$.
  Moreover, for $i \in \mathcal{T}_n \setminus \mathcal{L}_n$ we have

$$v_i^r = \max_{\pi \in i\mathcal{A}^\infty} \min_{m \in [1,M]} R_\pi^{T_m} = \max_{a \in \mathcal{A}} \max_{\pi \in ia\mathcal{A}^\infty} \min_{m \in [1,M]} R_\pi^{T_m} = \max_{a \in \mathcal{A}} v_{ia}^r \tag{6}$$

- The robust u-value u_i^r of a leaf node $i \in \mathcal{L}_n$ is the worst-case discounted sum of rewards $r_t = r(s_{m,t}, i_t)$ from the root to i. It is then backed-up to the rest of the tree:

$$u_i^r(n) \stackrel{\text{def}}{=} \begin{cases} \min_{m \in [1,M]} \sum_{t=0}^{d-1} \gamma^t r_t & \text{if } i \in \mathcal{L}_n \,; \\ \max_{a \in \mathcal{A}} u_{ia}^r(n) & \text{if } i \in \mathcal{T}_n \setminus \mathcal{L}_n \end{cases} \tag{7}$$

- Likewise, the robust b-value b_i^r is defined at leaf nodes and backed-up to the rest of the tree:

$$b_i^r(n) \stackrel{\text{def}}{=} \begin{cases} u_i^r(n) + \frac{\gamma^d}{1-\gamma} & \text{if } i \in \mathcal{L}_n \,; \\ \max_{a \in \mathcal{A}} b_{ia}^r(n) & \text{if } i \in \mathcal{T}_n \setminus \mathcal{L}_n \end{cases} \tag{8}$$

  An illustration of the computation of the robust b-values is presented in Figure 1.

Remark 1 (On the ordering of min and max). *In the definition of $u_i^r(n)$ it is essential that the minimum among the models is only taken at the end of trajectories, in the same way as for the robust objective* (3) *in which the worst-case dynamics is only determined after the policy has been fully specified. Assume that $u_i^r(n)$ is instead naively defined as:*

$$u_i^r(n) = \min_{m \in [1,M]} u_i^m(n),$$

This would not recover the robust policy, as we show in Figure 3 with a simple counter-example.

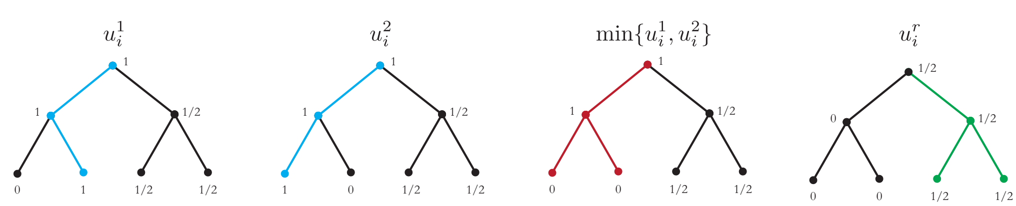


Figure 3: From left to right: two simple models and corresponding u-values with optimal sequences in blue; the naive version of the robust values returns sub-optimal paths in red; our robust u-value properly recovers the robust policy in green.

Algorithm 1: Deterministic Robust Optimistic Planning

Initialize $\mathcal{T}$ to a root and expand it. Set $n = 1$.
while *Numerical resource available* **do**
  Compute the robust u-values $u_i^r(n)$ and robust b-values $b_i^r(n)$.
  Expand $\text{argmax}_{i\in\mathcal{L}_n} b_i^r(n)$.
  n = n + 1
return $\text{argmax}_{a\in\mathcal{A}} u_a^r(n)$

From these definitions we introduce Algorithm 1, and analyse its sample-efficiency in Theorem 1.

Lemma 1 (Robust values ordering)**.** *The robust values, u-values and b-values exhibit similar properties as the optimal values, u-values and b-values, that is: for all* $0 < t < n$ *and* $i \in \mathcal{T}_n$*,*

$$u_i^r(t) \leq u_i^r(n) \leq v_i^r \leq b_i^r(n) \leq b_i^r(t) \tag{9}$$

Proof. This result stems directly from the definitions, see more details in Appendix A.1. ■

The simple regret of the action a returned by Algorithm 1 after n rounds is defined as:

$$\mathcal{R}_n = v^r - v_a^r \tag{10}$$

We will say that $\mathcal{R}_n = O(\varepsilon)$ for some $\varepsilon > 0$ if there exist $\rho > 0$ and $n_0 > 0$ such that $\mathcal{R}_n \leq \rho\varepsilon$ for all $n \geq n_0$. A node $i \in \mathcal{T}$ is said to be ϵ-optimal, in a robust sense, if and only if $v_i^r \geq v^r - \epsilon$ for some $\epsilon > 0$. The proportion of ϵ-optimal nodes at depth d is then defined as $p_d(\epsilon) = |i \in \mathcal{A}^d$ s.t i is ϵ-optimal$|/K^d$. Further we will assume that for the graph $\mathcal{T}$ the following hypothesis is satisfied:

Assumption 2 (Proportion of near-optimal nodes)**.** *There exist* $\beta \in [0, \frac{\log K}{\log 1/\gamma}]$, $c > 0$ *and* $d_0 > 0$ *such that* $p_d(\epsilon) \leq c\epsilon^\beta$ *for all* $\epsilon > 0$ *and* $d \geq d_0$.

Theorem 1 (Regret bound)**.** *Let* $\kappa = K\gamma^\beta \in [1, K]$. *Then the simple regret of Algorithm 1 is:*

$$\text{If } \kappa > 1, \qquad \mathcal{R}_n = O\left(n^{-\frac{\log 1/\gamma}{\log \kappa}}\right) \tag{11}$$

$$\text{If } \kappa = 1, \qquad \mathcal{R}_n = O\left(\gamma^{\frac{(1-\gamma)^\beta}{c} n}\right) \tag{12}$$

Proof. We use the properties shown in Lemma 1 and derive a robust counterpart of the proof of Hren and Munos (2008), which we only modify slightly. See more details in Appendix A.2 ■

3 Interval predictors

In this section, we assume that the ambiguity set Θ is continuous and bounded.

In the robust objective (3), the $\min$ operator only requires us to describe the set of states that can be reached with non-zero probability.

Definition The **reachability set** S at time t is the set of all states that can be reached by starting from initial state $s_0 \in \mathcal{S}$ and following a policy $\pi \in \mathcal{A}^\mathcal{S}$ along the transition dynamics $T_\theta \in \mathcal{S}^{\mathcal{S}\times\mathcal{A}}$.

$$S(t, s_0, \pi) \stackrel{\text{def}}{=} \{s_t : \exists\theta \in \Theta \text{ s.t. } s_{k+1} = T_\theta(s_k, a_k), a_k = \pi(s_k), k = 0, \cdots, t-1\} \tag{13}$$

This set can still have a complex shape. We approximate it by an overset easier to represent and manipulate: its interval hull.

Definition The **interval hull** of S, denoted $\square S = [\underline{s}, \overline{s}]$ is the smallest interval containing it:

$$\underline{s}(t, s_0, \pi) \stackrel{\text{def}}{=} \min S(t, s_0, \pi) \qquad \overline{s}(t, s_0, \pi) \stackrel{\text{def}}{=} \max S(t, s_0, \pi) \tag{14}$$

The max and min operators are applied element-wise. This set is illustrated in Figure 2.

State intervals $\square S$ have been used to describe the evolution of uncertain systems and derive feedback laws that achieve closed-loop stability in the presence of bounded disturbances (Stinga and Bunciu, 2012; Efimov and Raïssi, 2016; Dinh and Ito, 2017).

The main techniques of interval simulation have been listed and described in a survey by Puig et al. (2005), in which they are sorted into two categories. Region-based methods use the estimate of $\square S$

Algorithm 2: Interval-based Robust Control

Algorithm `robust_control`(s_0)
  Initialize a set Π of policies
  while *resources available* **do**
    `evaluate()` each policy $\pi \in \Pi$ at current state s_0
    Update Π by policy search
  end
  return $\operatorname{argmax}_{\pi\in\Pi} \hat{v^r}(\pi)$
Procedure `evaluate`(π, s_0)
  Compute the state interval $\square S(t, s_0, \pi)$ on a horizon $t \in [0, H]$
  Minimize r over the intervals $\square S(t, s_0, \pi)$ for all $t \in [0, H]$
  return $\hat{v^r}(\pi)$

at previous timestep $t-1$ to bootstrap the current estimate at time t. They are based on application of the theory of positive systems, which are frequently computationally efficient. However, the positive inclusion dynamics of a system may lead to overestimations of the true $\square S$ and even unstable behaviour. Trajectory-based methods estimate $\square S$ by taking the $\max$ and $\min$ in (14) over sampled trajectories for $\theta \in \Theta$. These methods produce subset estimates of the true $\square S$, do not suffer from the wrapping effect, but are often more computationally costly.

In this work, we leverage them to derive a proxy for the robust objective (3).

Definition Let us denote the robust objective of equation (3) as $v^r(\pi) \stackrel{\text{def}}{=} \min_{\theta\in\Theta} v_\pi^{T_\theta}$.
We define the **surrogate objective** $\hat{v^r}$ on a finite horizon $H > 0$ as:

$$\hat{v^r}(\pi) \stackrel{\text{def}}{=} \sum_{t=0}^{H} \gamma^t \min_{s\in\square S(t,s_0,\pi)} r(s, \pi(s)) \tag{15}$$

Property 1 (Lower bound)**.** *The surrogate objective $\hat{v^r}$ is a lower bound of the true objective v^r:*

$$\forall \pi, \hat{v^r}(\pi) \leq v^r(\pi) \tag{16}$$

Proof. By bounding the collected rewards by their minimum over $\square S(t)$. See Appendix A.3 ■

The robust objective error $v^r - \hat{v^r}$ stems from two terms: the interval approximation of the reachable set and the loss of time-dependency between the states within a single trajectory. If both these approximations are tight enough, maximizing the lower bound $\hat{v^r}$ will increase the true objective v^r, which is the idea behind Algorithm 2. It is classically structured as an alternation of a Policy Evaluation step , during which the surrogate objective $\hat{v^r}(\pi)$ is evaluated for a set of policies Π, and a Policy Search step which aims to steer the set of policies Π towards regions where the surrogate objective is maximal. The main Policy Search algorithms are listed in a survey by Deisenroth (2011). In this case, derivative-free methods such as evolutionary strategies (e.g. CMAES) would be more appropriate than policy gradient methods, since $\hat{v^r}$ cannot be easily differentiated. Planning algorithms can also be used to exploit the dynamics and structure of the surrogate objective.

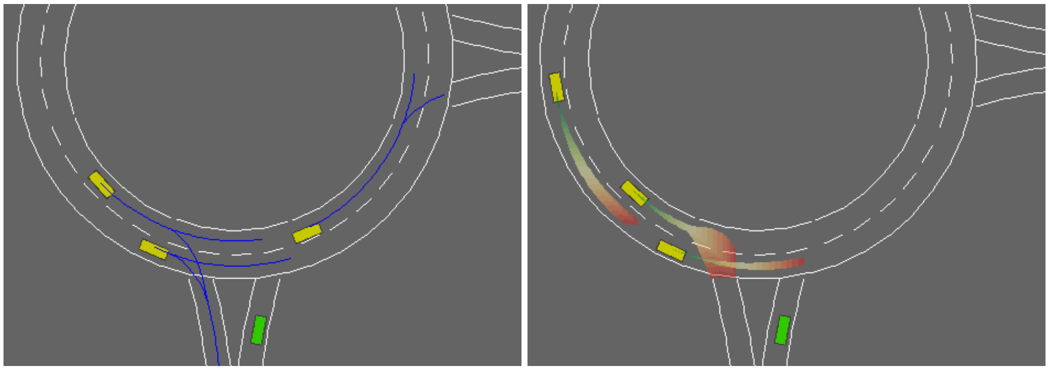
4 Experiments

Most autonomous driving architectures perform sequentially the prediction of other drivers' trajectories and the planning of a collision-free path for the ego-vehicle. As a consequence, they fail to account for interactions between the traffic participants and the ego-vehicle, leading to overly conservative decisions and a lack of negotiation abilities (Trautman and Krause, 2010). In this work, we perform both tasks *jointly* to anticipate the effect of our own decisions on the dynamics of the nearby traffic. But human decisions are not fully predictable and cannot be reduced to a single deterministic model. To avoid model bias, we provide a whole ambiguity set of reasonable closed-loop behavioural models for other vehicles, and plan robustly with respect to this ambiguity.

We introduce a new environment for simulated highway driving and tactical decision-making.[1]

Vehicle motion is described by the Kinematic Bicycle Model (see, e.g. Polack et al., 2017). They follow a lane keeping lateral behaviour, and a longitudinal behaviour inspired by the Intelligent Driver Model (Treiber et al., 2000) which balances reaching a desired velocity and respecting a safe time

[1]Source code is available at https://github.com/eleurent/highway-env



(a) The possible trajectories (blue) for fixed behaviours and varying destinations

(b) The possible trajectories (green-red gradient) for fixed destination and varying behaviours

Figure 4: The highway-env environment. The ego-vehicle (green) is approaching a roundabout with flowing traffic (yellow).

Table 1: Performances of robust planners on two ambiguous environments.

Ambiguity set	Agent	Worst-case return	Mean return $\pm$ std
True model	Oracle	9.83	10.84 ± 0.16
Discrete	Nominal	2.09	8.85 ± 3.53
	Algorithm 1	8.99	10.78 ± 0.34
Continuous	Nominal	1.99	9.95 ± 2.38
	Algorithm 2	7.88	10.73 ± 0.61

gap. The lane-change decisions are determined by the MOBIL model (Kesting et al., 2007): they must increase the vehicles accelerations while satisfying safe braking decelerations. The behaviour parameters θ of each traffic participant are sampled uniformly from a set Θ.

The ego-vehicle can be controlled with a finite set of tactical decisions $\mathcal{A} = \{$`no-op`, `right-lane`, `left-lane`, `faster`, `slower`$\}$ implemented by low-lever controllers. It is rewarded for driving fast along a planned route while avoiding collisions. More information on the environment modelling is provided in the appendices.

We carry out two experiments[2]: First, the behavioural parameters of traffic participants are fixed but their planned routes are unknown: we enumerate every direction they can take at their next intersection (see Figure 4a) and plan robustly with respect to this finite ambiguity set using Algorithm 1. Second, we assume on the contrary that the agents' planned routes are known but not their behavioral parameters (see Figure 4b). We plan robustly with respect to this continuous ambiguity set using Algorithm 2. Crucially, the state intervals prediction is conditioned on the planned policy π.

In both experiments, we compare the performance of the robust planner to an oracle model that has perfect knowledge of the systems dynamics, and to a nominal planner that plans optimistically with respect to a dynamics model sampled uniformly from the ambiguity set. Statistics are collected from 100 episodes with random environment initialization. Results are presented in Table 1.

5 Conclusion

This paper has presented two methods for approximately solving the robust control problem. In the simpler case of finite ambiguity set and action space, we use optimistic planning and provide an upper bound for the simple regret. A direct consequence is that we recover the robust policy as the computational budget increases. In the general case, we use interval prediction to efficiently solve a conservative approximation of the robust objective while providing a lower bound for the performance of a policy when applied to the unknown true model. However, this method is lossy and does not enjoy asymptotic consistency. Both algorithms are flexible, allowing to handle a variety of parametrized dynamical systems, and practical, with a focus on computational efficiency. The two methods are also orthogonal, which means they can be combined to deal with complex ambiguity sets that display both continuous and discrete features, such as disjoint unions of connected sets.

[2] Video and source code are available at https://eleurent.github.io/robust-control/

References

C.G. Atkeson and J.C. Santamaria. A comparison of direct and model-based reinforcement learning. *Proceedings of International Conference on Robotics and Automation*, 1997.

T. Basar and P. Bernhard. *H infinity - Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*, volume 41. 1996.

Sébastien Bubeck and Rémi Munos. Open Loop Optimistic Planning. Technical report, 2010.

Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. pages 72–83, 2006.

Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the Sample Complexity of the Linear Quadratic Regulator. 2017.

Marc Peter Deisenroth. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, 2011.

Thach Ngoc Dinh and Hiroshi Ito. Decentralization of Interval Observers for Robust Controlling and Monitoring a Class of Nonlinear Systems. 10:117–123, 2017.

D. Efimov and T. Raïssi. Design of interval observers for uncertain dynamical systems. *Automation and Remote Control*, 77:191–225, 2016.

Javier García and Fernando Fernández. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015. ISSN 15337928.

Jean-François Hren and Rémi Munos. Optimistic planning for deterministic systems. 2008.

Garud N. Iyengar. Robust Dynamic Programming. *Mathematics of Operations Research*, 30:257–280, 2005.

Arne Kesting, Martin Treiber, and Dirk Helbing. General Lane-Changing Model MOBIL for Car-Following Models. *Transportation Research Record: Journal of the Transportation Research Board*, pages 86–94, 2007.

Hassan K. Khalil. *Nonlinear Control*. Pearson, 2014. ISBN 013349926X.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. pages 282–293, 2006.

Shihua Li, Xinghuo Yu, Leonid Fridman, Zhihong Man, and Xiangyu Wang. *Advances in Variable Structure Systems and Sliding Mode Control – Theory and Applications (Studies in Systems, Decision and Control)*. Springer, 2017.

Odalric-Ambrym Maillard. Self-normalization techniques for streaming confident regression. 2017.

Arnab Nilim and Laurent El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research*, 53:780–798, 2005.

Ian R. Petersen and Roberto Tempo. Robust control of uncertain systems: Classical results and recent developments. *Automatica*, 50(5):1315–1335, 2014.

Philip Polack, Florent Altché, and Brigitte D'Andréa-Novel. The Kinematic Bicycle Model : a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles ? pages 6–8, 2017.

Vicenç Puig, Alexandru Stancu, and Joseba Quevedo. *Simulation of Uncertain Dynamic Systems Described By Interval Models: a Survey*, volume 38. IFAC, 2005.

JG Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. *Advances in neural information processing systems*, pages 1047—-1053, 1997.

F. Stinga and E. Bunciu. Robust interval observer and nonlinear predictive control of an active sludge process. *System Theory, Control and Computing*, (1), 2012.

Richard S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In *Machine Learning Proceedings 1990*. 1990.

Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010.

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 62(2):1805–1824, 2000.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov Decision Processes. … *of Operations Research*, pages 1–52, 2013.

Supplementary material

A Detailed proofs

A.1 Lemma 1

Proof. By definition, when starting with sequence i, the value $u_i^m(n)$ represents the minimum admissible reward, while $b_i^m(n)$ corresponds to the best admissible reward achievable with respect to the the possible continuations of i. Thus, for all $i \in \mathcal{A}^*$, $u_i^m(n)$ and $u_i^r(n)$ are non-decreasing functions of n and $b_i^m(n)$ and $b_i^r(n)$ are a non-increasing functions of n, while v_i^m and v_i^r do not depend on n.

Moreover, since the reward function r is assumed to have values in $[0, 1]$, the sum of discounted rewards from a node of depth d is at most $\gamma^d + \gamma^{d+1} + \dots = \frac{\gamma^d}{1-\gamma}$. As a consequence, for all $n \geq 0$, $i \in \mathcal{L}_n$ of depth d, and any sequence of rewards $(r_t)_{t\in\mathbb{N}}$ obtained from following a path in $i\mathcal{A}^\infty$ with any dynamics $m \in [1, M]$:

$$\sum_{t=0}^{d-1} \gamma^t r_t \leq \sum_{t=0}^{d-1} \gamma^t r_t + \sum_{t=d}^{\infty} \gamma^t r_t \leq \sum_{t=0}^{d-1} \gamma^t r_t + \frac{\gamma^d}{1-\gamma}$$

That is equivalent to:

$$u_i^m(n) \leq \sum_{t=0}^{\infty} \gamma^t r_t \leq b_i^m(n)$$

Hence,

$$\min_{m\in[1,M]} u_i^m(n) \leq \min_{m\in[1,M]} \sum_{t=0}^{\infty} \gamma^t r_t \leq \min_{m\in[1,M]} b_i^m(n) \tag{17}$$

And as the left-hand and right-hand sides of (17) are independent of the particular path that was followed in $i\mathcal{A}^\infty$, it also holds for the robust path:

$$\min_{m\in[1,M]} u_i^m(n) \leq \max_{\pi\in i\mathcal{A}^\infty} \min_{m\in[1,M]} \sum_{t=0}^{\infty} \gamma^t r_t \leq \min_{m\in[1,M]} b_i^m(n)$$

that is,

$$u_i^r(n) \leq v_i^r \leq b_i^r(n) \tag{18}$$

Finally, (18) is extended to the rest of $\mathcal{T}_n$ by recursive application of (6), (7) and (8). ■

A.2 Theorem 1

Proof. Hren and Munos (2008) first show in Theorem 2 that the simple regret of their optimistic planner is bounded by $\frac{\gamma^{d_n}}{1-\gamma}$ where d_n is the depth of $\mathcal{T}_n$. This properties relies on the fact that the returned action belongs to the deepest explored branch, which we can show likewise by contradiction using Lemma 1. This yields directly that $a = i_0$ where i is some node of maximal depth d_n expanded at round $t \leq n$, which by Algorithm 1 verifies $b_a^r(t) = b_i^r(t) = \max_{x\in\mathcal{A}} b_x^r(t)$ and:

$$v^r - v_a^r = v_{a^*}^r - v_a^r \leq b_{a^*}^r(t) - v_a^r \leq b_a^r(t) - u_a^r(t) = b_i^r(t) - u_i^r(t) = \frac{\gamma^{d_n}}{1-\gamma} \tag{19}$$

Secondly, they bound the depth d_n of $\mathcal{T}_n$ with respect to n. To that end, they show that the expanded nodes always belong to the sub-tree $\mathcal{T}_\infty$ of all the nodes of depth d that are $\frac{\gamma^d}{1-\gamma}$-optimal. Indeed, if a node i of depth d is expanded at round n, then $b_i^r(n) \geq b_j^r(n)$ for all $j \in \mathcal{L}_n$ by Algorithm 1, thus the max-backups of (8) up to the root yield $b_i^r(n) = b_\emptyset^r(n)$. Moreover, by Lemma 1 we have that $b_\emptyset^r(n) \geq v_\emptyset^r = v^r$ and so $v_i^r \geq u_i^r(n) = b_i^r(n) - \frac{\gamma^d}{1-\gamma} \geq v^r - \frac{\gamma^d}{1-\gamma}$, thus $i \in \mathcal{T}_\infty$.

Then from Assumption 2 and the definition of β applied to nodes in $\mathcal{T}_\infty$, there exists d_0 and c such that the number n_d of nodes of depth $d \geq d_0$ in $\mathcal{T}_\infty$ is bounded by $c\left(\frac{\gamma^d}{1-\gamma}\right)^\beta K^d$. As a consequence,

$$\begin{aligned} n &= \textstyle\sum_{d=0}^{d_n} n_d = n_0 + \sum_{d=d_0+1}^{d_n} n_d \\ &\leq n_0 + \textstyle\sum_{d=d_0+1}^{d_n} c\left(\frac{\gamma^d}{1-\gamma}\right)^\beta K^d \\ &= n_0 + c' \textstyle\sum_{d=d_0+1}^{d_n} \kappa^d \end{aligned}$$

where $c' = \frac{c}{(1-\gamma)^\beta}$.

- If $\kappa > 1$, then $n \le n_0 + c'\kappa^{d_0+1}\frac{\kappa^{d_n-d_0}-1}{\kappa-1}$ and thus $d_n \ge d_0 + \log_\kappa \frac{(n-n_0)(\kappa-1)}{c'\kappa^{d_0+1}}$. We conclude from (19) that $\mathcal{R}_n \le \frac{\gamma^{d_n}}{1-\gamma} = \frac{1}{1-\gamma}\left(\frac{(n-n_0)(\kappa-1)}{c'\kappa^{d_0+1}}\right)^{\frac{\log\gamma}{\log\kappa}} = O\left(n^{-\frac{\log 1/\gamma}{\log\kappa}}\right)$.
- If $\kappa = 1$, then $n \le n_0 + c'(d_n - d_0)$, hence from (19) we have $\mathcal{R}_n = O\left(\gamma^{nc'}\right)$.

■

A.3 Property 1

Proof. For any $\theta \in \Theta$, $t \in [0, H]$ and any trajectory $(s_0, \cdots, s_t)$ sampled from π and T_θ,

$$s_t \in S(t, s_0, \pi) \subset \square S(t, s_0, \pi)$$

Hence,

$$R_\pi^{T_\theta} = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \ge \sum_{t=0}^{H} \gamma^t r(s_t, a_t) \ge \sum_{t=0}^{H} \min_{s \in \square S(t, s_0, \pi)} \gamma^t r(s, \pi(s)) = \hat{v^r}(\pi)$$

And finally,

$$v^r(\pi) = \min_{\theta\in\Theta} v_\pi^{T_\theta} = \min_{\theta\in\Theta} \mathbb{E}(R_\pi^{T_\theta}) \ge \hat{v^r}(\pi)$$

■

B Environment dynamics

B.1 Kinematics

The vehicles kinematics are represented by the Kinematic Bicycle Model:

$$\dot{x} = v\cos(\psi), \tag{20}$$
$$\dot{y} = v\sin(\psi), \tag{21}$$
$$\dot{v} = a, \tag{22}$$
$$\dot{\psi} = \frac{v}{l} tan(\beta), \tag{23}$$

where (x, y) is the vehicle position, v its forward velocity and ψ its heading, l is the vehicle half-length, a is the acceleration command and β is the slip angle at the center of gravity, used as a steering command.

Each vehicle i is represented by its kinematics $X_i = [x_i, y_i, v_i, \psi_i]$. The joint state is represented by $s = \{X_1, \cdots, X_N\}$

B.2 Longitudinal control

The acceleration control is assumed to be linearly parametrized:

$$a = \theta_a^T \phi_a(s, i), \tag{24}$$

where θ_a is an uncertain weight vector, and $\phi_a(s, i)$ is a feature vector that depends on the joint state s and considered vehicle i.

It is composed of:

- a target velocity seeking term,
- a braking term to adjust velocity w.r.t. the front vehicle ,
- a braking term to respect a safe distance w.r.t. the front vehicle.

Denoting f_i the front vehicle preceding vehicle i, ϕ_a is defined by

$$\phi_a(s,i) = \begin{bmatrix} v_0 - v_i \\ n(v_{f_i} - v_i) \\ n(x_{f_i} - x_i - (d_0 + v_i T)) \end{bmatrix} \tag{25}$$

where n is the negative part function $n(x) = \min(x, 0)$ and v_0, d_0 and T respectively denote the speed limit, jam distance and time gap given by traffic rules.

We observe that this model exhibits similar qualitative behaviours to the IDM's.

B.3 Lateral control

A non-linear lane-keeping controller is implemented as follows: a lane L with lateral position y_L and heading ψ_L is tracked by performing

1. Position control
$$v_{y_{cmd}} = K_{p_y}(y_L - y) \tag{26}$$
2. Lateral velocity to heading conversion
$$\psi_{ref} = \psi_L + \sin^{-1}\left(\frac{v_{y_{cmd}}}{v}\right) \tag{27}$$
3. Heading control
$$\psi_{cmd} = K_{p_\psi}(\psi_{ref} - \psi) \tag{28}$$
4. Heading rate to steering angle conversion
$$\beta = \tan^{-1}(\frac{l}{v}\psi_{cmd}) \tag{29}$$

Finally,

$$\beta = \tan^{-1}(\frac{l}{v}K_{p_\psi}(\psi_L + \sin^{-1}\left(K_{p_y}\frac{y_L - y}{v}\right) - \psi)) \tag{30}$$

This non-linear controller presented in subsection can be linearised around its equilibrium $(y, \psi) = (y_L, \psi_L)$.

$$\begin{aligned} \frac{l}{v}\tan\beta &= K_{p_\psi}(\psi_L + \sin^{-1}\left(K_{p_y}\frac{y_L - y}{v}\right) - \psi) & (31) \\ &\simeq \frac{l}{v}(K_{p_\psi}(\psi_L + \left(K_{p_y}\frac{y_L - y}{v}\right) - \psi)) & (32) \\ &= \theta_b^T \phi_b & (33) \end{aligned}$$

with

$$\theta_b = \begin{bmatrix} K_{p_\psi} & K_{p_y}K_{p_\psi} \end{bmatrix}^T \tag{34}$$

and

$$\phi_b = \begin{bmatrix} \psi_L - \psi \\ \frac{1}{v}(y_L - y) \end{bmatrix} \tag{35}$$

B.4 Discrete behaviour

The MOBIL model (Kesting et al., 2007), which stands for *Minimizing Overall Braking Induced by Lane Changes*, is a discrete lateral decision model that formulates a criterion for lane changes in terms of safe braking decelerations and increased overall accelerations according to a longitudinal model.

It states that a lane change should be performed if and only if:

1. It does not impose an unsafe braking on the target lane following vehicle:
$$\dot{v}_{\text{rear}} \geq -b_{\text{safe}} \tag{36}$$
2. It enables the vehicle and (with a politeness factor p) its following vehicles on both current and target lanes to increase their overall acceleration:
$$\Delta\dot{v} + p(\Delta\dot{v}_{\text{rear, current}} + \Delta\dot{v}_{\text{rear, target}}) \geq a_{min} \tag{37}$$

This model describes changes in the target lane L.

C Interval Predictor

In this section, we design an interval predictor for our system.

C.1 Notations

For any real variable z, we denote an interval containing z as $\square z = [\underline{z}, \overline{z}]$, such that $\underline{z} \leq z \leq \overline{z}$. As elements of $\mathbb{R}^2$, they can be scaled and offset by scalars. This definition is extended element-wise to vector variables.

Then, we define several operators over intervals $\square a = [\underline{a}, \overline{a}]$ and $\square b = [\underline{b}, \overline{b}]$

- The product operator $\times$

$$\square a \times \square b = [p(\underline{a})p(\underline{b}) - p(\overline{a})n(\underline{b}) - n(\underline{a})p(\overline{b}) + n(\overline{a})n(\overline{b}), \tag{38}$$

$$p(\overline{a})p(\overline{b}) - p(\underline{a})n(\overline{b}) - n(\overline{a})p(\underline{b}) + n(\underline{a})n(\underline{b})] \tag{39}$$

  where $p(\cdot)$ and $n(\cdot)$ are the projections onto $\mathbb{R}^+$ and $\mathbb{R}^-$, respectively.
- The difference operator $-$

$$\square a - \square b = [\underline{a} - \overline{b}, \overline{a} - \underline{b}] \tag{40}$$

- The cosine and sine operators

$$\cos(\square z) = [-1 \text{ if } \underline{z} \leq \pi \leq \overline{z} \text{ else } \min(\cos(\underline{z}), \cos(\overline{z})), \tag{41}$$

$$1 \text{ if } \underline{z} \leq 0 \leq \overline{z} \text{ else } \max(\cos(\underline{z}), \cos(\overline{z}))] \tag{42}$$

$$\sin(\square z) = [-1 \text{ if } \underline{z} \leq -\frac{\pi}{2} \leq \overline{z} \text{ else } \min(\sin(\underline{z}), \sin(\overline{z})), \tag{43}$$

$$1 \text{ if } \underline{z} \leq +\frac{\pi}{2} \leq \overline{z} \text{ else } \max(\sin(\underline{z}), \sin(\overline{z}))] \tag{44}$$

- The inverse operator $/$ over a positive interval $\square z > 0$

$$1/\square z = [1/\overline{z}, 1/\underline{z}] \tag{45}$$

- Any other function f is assumed increasing on the interval $\square z$ and is applied coefficient-wise

$$f(\square z) = [f(\underline{z}), f(\overline{z})] \tag{46}$$

We start with an initial estimate of the intervals over state variables x_I, y_I, v_I and ψ_I. Typically, we use zero-width intervals centred on the current state observation. Likewise, any variable z used in place of an interval corresponds to the zero-width interval $[z, z]$.

C.2 Intervals for features

We use (25) and (35) respectively to derive intervals for the features ϕ_a and ϕ_b from the intervals over the states.

We index the front vehicle intervals with the subscript f

$$\square \phi_a = \begin{bmatrix} v_0 - \square v \\ n(\square v_f - \square v) \\ n(\square x_f - \square x - (d_0 + T\square v)) \end{bmatrix} \tag{47}$$

and

$$\square \phi_b = \begin{bmatrix} (1 \;/\; \square v) \times (y_L - \square y) \\ \psi_L - \square \psi \end{bmatrix} \tag{48}$$

C.3 Intervals for controls

The controls intervals are derived from (24) and (33)

$$\square a = \square \theta_a^T \times \square \phi_a \tag{49}$$

$$\square \left(\frac{l}{v} \tan \beta\right) = \square \theta_b^T \times \square \phi_b \tag{50}$$

C.4 Intervals for velocity and heading

The velocity interval is derived from (22) and the heading interval from (23)

$$\Box\dot{v} = \Box a \tag{51}$$

$$\Box\dot{\psi} = \Box\left(\frac{l}{v}\tan\beta\right) \tag{52}$$

C.5 Intervals for positions

Likewise, the positions interval are derived from the kinematics (20) and (21)

$$\Box\dot{x} = \Box v \times \cos(\Box\psi) \tag{53}$$

$$\Box\dot{y} = \Box v \times \sin(\Box\psi) \tag{54}$$