FormulaZero: Distributionally Robust Online Adaptation via Offline Population Synthesis

Aman Sinha [*1] Matthew O'Kelly [*2] Hongrui Zheng [*2] Rahul Mangharam [2] John Duchi [1] Russ Tedrake [3]

Abstract

Balancing performance and safety is crucial to deploying autonomous vehicles in multi-agent environments. In particular, autonomous racing is a domain that penalizes safe but conservative policies, highlighting the need for robust, adaptive strategies. Current approaches either make simplifying assumptions about other agents or lack robust mechanisms for online adaptation. This work makes algorithmic contributions to both challenges. First, to generate a realistic, diverse set of opponents, we develop a novel method for self-play based on replica-exchange Markov chain Monte Carlo. Second, we propose a distributionally robust bandit optimization procedure that adaptively adjusts risk aversion relative to uncertainty in beliefs about opponents' behaviors. We rigorously quantify the tradeoffs in performance and robustness when approximating these computations in real-time motion-planning, and we demonstrate our methods experimentally on autonomous vehicles that achieve scaled speeds comparable to Formula One racecars.

1. Introduction

Current autonomous vehicle (AV) technology still struggles in competitive multi-agent scenarios, such as merging onto a highway, where both maximizing performance (negotiating the merge without delay or hesitation) and maintaining safety (avoiding a crash) are important. The strategic implications of this tradeoff are magnified in racing. During the 2019 Formula One season, the race-winner achieved the fastest lap in only 33% of events (Federation Internationale de l'Automobile, 2019). Empirically, the weak correlation between achieving the fastest lap-time and winning suggests that consistent and robust performance is critical to success. In this paper, we investigate this intuition in the setting of autonomous racing (AR). In AR, an AV must lap a racetrack in the presence of other agents deploying unknown policies. The agent wins if it completes the race faster than its opponents; a crash automatically results in a loss.

[*]Equal contribution [1]Stanford University, Stanford, CA, USA [2]University of Pennsylvania, Philadelphia, PA, USA [3]Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Correspondence to: Aman Sinha <amans@stanford.edu>, Matthew O'Kelly <mokelly@seas.upenn.edu>, Hongrui Zheng <hongruiz@seas.upenn.edu>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

AR is a competitive multi-agent game, a general setting challenging for a number of reasons, especially in robotics applications. First, failures are expensive and dangerous, so learning-based approaches must avoid such behavior or rely on simulation while training. Second, the agents only partially observe their opponent's state, and these observations do not uniquely determine the opponent's behavior. Finally, the agents must make decisions online; the opponent's strategy is a tightly-held secret and cannot be obtained by collecting data before the competition.

Problem: We frame the AR challenge in the context of robust reinforcement learning. We analyze the system as a partially-observed Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, P_{sa}, \mathcal{O}, r, \lambda)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, state-action transition probabilities P_{sa}, observation space $\mathcal{O}$, rewards $r : \mathcal{O} \to \mathbb{R}$, and discount factor λ. Furthermore, we capture uncertainty in behaviors of other agents through an *ambiguity*[1] set $\mathcal{P}$ for the state-action transitions. Then the AV's objective is

$$\text{maximize} \inf_{P_{sa} \in \mathcal{P}} \sum_t \lambda^t \mathbb{E}[r(o(t))]. \tag{1}$$

The obvious price of robustness (Bertsimas & Sim, 2004) is that a larger ambiguity set ensures a greater degree of safety while sacrificing performance against a particular opponent. If we knew the opponent's behavior, we would need no ambiguity set; equivalently, the ambiguity set would shrink to the nominal state-action transition distribution. Our goal is to automatically trade between performance and robustness as we play against opponents, which breaks down into two challenges: parametrizing the ambiguity set to allow tractable inference and computing the robust cost efficiently online.

[1]Ambiguity is a synonym for uncertainty (Gilboa & Marinacci, 2016). Formal descriptions in this paper use the term ambiguity.

Contributions: The paper has three contributions: (i) a novel population-based self-play method to parametrize opponent behaviors, (ii) a provably efficient approach to estimate the ambiguity set and the robust cost online, and (iii) a demonstration of these methods on real autonomous vehicles. The name of our approach—FormulaZero—alludes both to the Formula One racing league and the fact that we use self-play (and no demonstrations) to learn competitive behaviors, similar to the approach of AlphaZero (Silver et al., 2018).

Section 1.1 gives context to our learning problem, including connections to classical control techniques. In Section 2, we describe the first challenge: learning how to parametrize the ambiguity set $\mathcal{P}$. Rather than directly consider the continuous action space of throttle and steering outputs, we synthesize a library of "prototype" opponent behaviors offline using population-based self-play. When racing against a particular opponent, the agent maintains a belief vector $w(t)$ of the opponent's behavior patterns as a categorical distribution over these prototype behaviors. We then parametrize the ambiguity set as a ball around this nominal belief $w(t)$.

The second challenge, presented in Section 3, is an online optimization problem, wherein the agent iteratively updates the ambiguity set (*e.g.* updates $w(t)$) and computes the robust cost of this set. In other words, the agent attempts to learn the opponent's behavior online to maximize its competitive performance. Since this optimization occurs on a moving vehicle with limited computational resources, we provide convergence results that highlight tradeoffs of performance and robustness with respect to these budgets. Finally, Section 4 details the practical implications of the theoretical results, emergent properties of the method, and the experimental performance of our approach.

1.1. Related work

Reinforcement learning (RL) has achieved unprecedented success on classic two-player games (*e.g.* Silver et al., 2018), leading to new approaches in partially-observable games with continuous action spaces (Arulkumaran et al., 2019; Berner et al., 2019). In these works, agents train via self-play using Monte Carlo tree search (Browne et al., 2012; Sutton & Barto, 2018) or population-based methods (Jaderberg et al., 2017; 2019). The agents optimize expected performance rather than adapt to individual variations in opponent strategy, which can lead to poor performance against particular opponents (Bansal et al., 2017). In contrast, our method explicitly incorporates adaptivity to opponents.

Robust approaches to RL and control (like this work) explicitly model uncertainty. In RL, this amounts to planning in a robust MDP (Nilim & El Ghaoui, 2005) or a POMDP (Kaelbling et al., 1998). Early results Bagnell et al. (2001) and Nilim & El Ghaoui (2005) describe solutions for robust planning in (PO)MDPs with tabular state/action spaces. Equivalent results in control are analytical formulations applicable to uncertainty in linear time-invariant systems (Doyle et al., 1988; Vinnicombe, 1993; Zhou et al., 1996). Recent works (Tamar et al., 2014; Pinto et al., 2017; Mandlekar et al., 2017; Gleave et al., 2019) describe minimax and adversarial RL frameworks for nonlinear systems and continuous action spaces. Like our approach, these methods fall broadly under the framework of robust optimization. Unlike these works, which consider worst-case planning under a fixed uncertainty distribution, our approach updates the distribution online.

Our approach is designed to adjust the agent's evaluation of short-term plans relative to uncertainty in the opponent's behavior rather than provide worst-case guarantees. Complementary to and compatible with our approach are techniques which provide the latter guarantees, such as robust model predictive control (Bemporad & Morari, 1999). Extensions of robust control for nonlinear systems and complex uncertainty models are also compatible (*e.g.* Majumdar & Tedrake (2013); Althoff & Dolan (2014); Gao et al. (2014)). In contrast to formal approaches which explicitly guarantee robustness, some authors have proposed multitask or meta-learning approaches (*e.g.* Caruana (1997); He et al. (2016); Finn et al. (2018)) can implicitly learn to play against multiple opponents. However, such techniques do not explicitly model uncertainty or quantify robustness, which we deem necessary in the high-risk, safety-critical regime.

Planning in belief space is closely related to our approach and is well-studied in robotics (see *e.g.* Kochenderfer, 2015). Specifically in the AV domain, Galceran et al. (2015) and Ding & Shen (2019) use a Bayesian approach to plan trajectories for AVs in belief space; like this work, both of these approaches characterize the other agent's behavior in the environment categorically. Also similar to this work, Van Den Berg et al. (2011) use a sampled set of goals obtained by planning from other agents' perspectives. The main difference in this work from standard belief-space planning formulations is inspired by recent results from distributionally robust optimization (DRO) in supervised-learning settings (Ben-Tal et al., 2013; Namkoong & Duchi, 2017). These methods reweight training data to reduce the variance of the training loss (Namkoong & Duchi, 2017). While others apply DRO to episodic RL for training *offline* (Sinha et al., 2017; Smirnova et al., 2019), we reweight the belief *online*.

Online methods for control fall under the umbrella of adaptive control (Kumar, 1985; Åström & Wittenmark, 2013). Dean et al. (2018) and Agarwal et al. (2019) establish regret bounds for adaptive control methods applied to LTI systems, tightening the relationship to online learning. Due to the more general nature of our problem, we draw from the adversarial multi-armed bandit framework of online learning (Abernethy & Rakhlin, 2009; Bubeck et al., 2012; Shalev-Shwartz et al., 2012).

Our belief state corresponds to a categorical distribution of polices governing an opponent's next action; the goal is to predict which strategy the opponent is using and compute the best response. This approach is similar to game-theoretic methods for AR and AV decision making that use the standard heuristic of iterated best response. Our work is distinct from previous work, which either assumes that all agents act with respect to the same cost function, simplifying the structure of the game (Liniger & Lygeros, 2019; Wang et al., 2019); or, without this simplifying assumption, that uses demonstrations to learn possible sets of policies (Sadigh et al., 2016; Williams et al., 2017). In constrast, we learn the set of policies without demonstrations and use DRO to robustly score the AV's plans.

We convert the problem of predicting opponent behavior in a continuous action space into an adversarial bandit problem by learning a set of cost functions that characterize a discrete set of policies. As a result, we would like the opponent models to be both near-optimal and diverse. We use determinantal point processes (DPPs) (Kulesza et al., 2012) to sample diverse configurations of the parameter space. However, first we must learn a DPP kernel, which requires that we efficiently sample *competitive* cost functions from the larger configuration space. Since we assume no structure to the set of competitive cost functions, we employ a Markov-chain Monte Carlo (MCMC) method. Complementary methods include variational-inference (*e.g.* Arenz et al. (2018)) and evolutionary (*e.g.* Mouret & Clune (2015)) approaches, which can be challenging to scale up to unstructured, high-dimensional settings of which we have little prior domain knowledge. In our approach, we update the classic simulated tempering method (Marinari & Parisi, 1992) with a novel annealing scheme (Kirkpatrick et al., 1983; Černỳ, 1985) designed for population diversity. We describe this approach next.

2. Offline population synthesis

The goal of offline population synthesis is to generate a diverse set of competitive agent behaviors. Formally, we would like to sample pairs $(x, \theta) \in \mathcal{X} \times \Theta$ that are both diverse as well as achieve small values for a function $f(x, \theta)$. In our AV application, θ parametrizes a neural network used to sample trajectories to follow, x is a weighting of various cost functions that the vehicle uses to select trajectories from the samples, and f is the simulated lap time. With this motivation, we treat the method in more generality assuming (as in our application) that while we can differentiate $f(x, \theta)$ with respect to θ, x represents hyperparameters and admits only function evaluations $f(x, \theta)$ rather than first-order developments. The key challenge is that we do not *a priori* know a metric with which to evaluate diversity (*e.g.*, a kernel for a DPP) nor do we know a base value of f that is deemed acceptable for competitive performance.

We make this problem more tractable via temperature-based Markov-chain Monte Carlo (MCMC) and annealing methods (Matyas, 1965; Hastings, 1970; Kirkpatrick et al., 1983; Černỳ, 1985; Ingber, 1993; Hu & Hu, 2011). Our goal is to sample from a Boltzmann distribution $g(x, \theta; \beta(t)) \propto e^{-\beta(t) f(x,\theta)}$, where $\beta(t)$ is an inverse "temperature" parameter that grows (or "anneals") with iterations t. When $\beta(t) = 0$, all configurations (x, θ) are equally likely and all MCMC proposals are accepted; as $\beta(t)$ increases, accepted proposals favor smaller f. Unlike standard hyperparameter optimization methods (Bergstra & Bengio, 2012; Jaderberg et al., 2017) that aim to find a single near-optimal configuration, our goal is to sample a diverse population of (x, θ) achieving small $f(x, \theta)$. As such, our approach—annealed adaptive population tempering (AADAPT)—maintains a population of configurations and employs high-exploration proposals based on the classic hit-and-run algorithm (Smith, 1984; Bélisle et al., 1993; Lovász, 1999).

2.1. AADAPT

AADAPT builds upon replica-exchange MCMC, also called parallel tempering, which is a standard approach to maintaining a population of configurations (Swendsen & Wang, 1986; Geyer, 1991). In parallel tempering, one maintains replicas of the system at L different temperatures $\beta_1 \geq \beta_2 ... \geq \beta_L$ (which are predetermined and fixed), defining the density of the total configuration as $\prod_{i=1}^{L} g(x^i, \theta^i; \beta_i)$. The configurations at each level perform standard MCMC steps (also called "vertical" steps) as well as "horizontal" steps wherein particles are swapped between adjacent temperature levels (see Figure 1). Horizontal proposals consist of swapping two configurations in adjacent temperature levels uniformly at random; the proposal is accepted using standard Metropolis-Hastings (MH) criteria (Hastings, 1970). The primary benefit of maintaining parallel configurations is that the configurations at "colder" levels (higher β) can exploit high-exploration moves from "hotter" levels (lower β) which "tunnel" down during horizontal steps (Geyer, 1991). This approach allows for faster mixing times, particularly when parallel MCMC proposals occur concurrently in a distributed computing environment.

Maintaining a population: In AADAPT (Algorithm 1), we maintain a *population* of D configurations at each separate temperature level. Note that this design always maintains D individuals at the highest performance level (highest β). The overall configuration density is $\prod_{i=1}^{L} \prod_{j=1}^{D} g(x^{i,j}, \theta^{i,j}; \beta_i(t))$. Similar to parallel tempering, horizontal proposals are chosen uniformly at ran-

Algorithm 1 AADAPT

```
input: annealing parameter $\alpha$, vertical steps $V$, horizontal exchange steps $E$, temperature levels $L$, population size $d$, initial samples $\{x^{i,j}, \theta^{i,j}\}_{i\in\{1,L\}}^{j\in\{1,D\}}$, iterations $T$
Evaluate $f(x^{i,j}, \theta^{i,j})$
for $t = 1$ to $T$
  for $j = 1$ to $L$ do anneal $\beta_{L-j+1}(t)$ (problem (2))
  for $k = 1$ to $V$ asynchronously, in parallel
    for each population $i$ asynchronously, in parallel
      Sample $\hat{x}^{i,j}$ according to hit-and-run proposal
      Evaluate $f(\hat{x}^{i,j}, \theta^{i,j})$
      Apply MH criteria to update $x^{i,j}$
      Train $\theta^{i,j}$ via SGD
  for $e = 1$ to $E$ do horizontal swaps (Appendix A)
```

dom from configurations at adjacent temperatures (see Appendix A). We get the same computational benefits of fast mixing in distributed computing environments and a greater ability to exploit high-temperature "tunneling" due to the greater number of possible horizontal exchanges between adjacent temperature levels. The benefit of the horizontal steps is even more pronounced in the RL setting as only vertical steps require new evaluations of f (*e.g.* simulations).

High-exploration vertical proposals: Another benefit of maintaining parallel populations is to improve exploration. We further improve exploration by using hit-and-run proposals (Smith, 1984; Bélisle et al., 1993; Lovász, 1999) for the vertical MCMC chains. Namely, from a current point (x, θ) we sample a uniformly random direction $\hat{u}$ and then choose a point uniformly on the segment $\mathcal{X} \cap (\{x + \mathbb{R}\cdot\hat{u}\} \times \{\theta\})$. This approach has several guarantees for efficient mixing (Lovász, 1999; Lovász & Vempala, 2003; 2006). Note that in our implementation the MCMC steps are only performed on x, while θ updates occur via SGD (see below).

Adaptively annealed temperatures: A downside to parallel tempering is the need to determine the temperature levels β_i beforehand. In AADAPT. we adaptively update temperatures. Specifically, we anneal the prescribed horizontal acceptance probability of particle exchanges between temperature levels as $\alpha^{t/(L-1)}$ for a fixed hyperparameter $\alpha \in (0, 1)$. Define the empirical acceptance probability of swaps of configurations between levels $i-1$ and i as

$$p_{i-1,i} := \frac{1}{D^2}\sum_{j=1}^{D}\sum_{k=1}^{D}(y_{i-1,i}^{j,k})^{\beta_{i-1}-\beta_i}$$

$$y_{i-1,i}^{j,k} := \min\left(1, e^{f(x^{i-1,j},\theta^{i-1,j})-f(x^{i,k},\theta^{i,k})}\right).$$

Then, at the beginning of each iteration (in which we perform a series of vertical and horizontal MCMC steps), we update the $\beta_i(t)$ sequentially; we fix $\beta_L(t) := \beta_L = 0$ and for a given β_i, we set β_{i-1} by solving the following convex optimization problem:

$$\underset{\{\beta_{i-1}\geq\beta_i,\ \ p_{i-1,i}\leq\alpha^{\frac{t}{(L-1)}}\}}{\text{minimize}} \quad \beta_{i-1}, \tag{2}$$

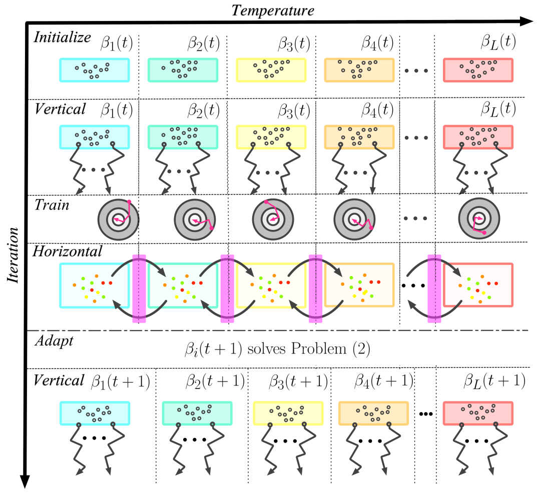


Figure 1. Illustration of AADAPT. Vertical MCMC steps (jagged black arrows) occur in parallel for $x^{i,j}$, followed by gradient descent for trainable parameters $\theta^{i,j}$ (magenta arrows) and horizontal MCMC configuration swaps between populations (curved black arrows). Temperatures $\beta_i(t)$ are then updated by problem (2).

using binary search. This adaptive scheme is crucial in our problem setting, where we *a priori* have no knowledge of appropriate scales for f and, as a result, β. In practice, we find that forcing β_i to monotonically increase in t yields better mixing, so we set $\beta_i(t) = \max(\beta_i(t-1), \hat{\beta}_i(t))$, where $\hat{\beta}_i(t)$ solves problem (2).

Evaluating proposals via self-play: We apply AADAPT to a multi-agent game. It is only possible to evaluate $f(x, \theta)$ in the context of other agents, but we consider the setting where demonstrations from potential opponents are either difficult to obtain or held secret. Thus, we iteratively evaluate f via self-play. For each configuration (x, θ), we perform a race in the simulated environment between two vehicles with the same policy (with $f(x, \theta)$ being the lap time of the agent that starts behind the other). Vertical MCMC steps propose new x, which are then accepted according to MH criteria. After a number of vertical iterations, a stochastic gradient descent (SGD) step is applied to θ (which maximizes the likelihood of the trajectories chosen by the agent with cost functions parametrized by x). Following this process, the updated agents in adjacent temperature levels are exchanged via horizontal MCMC steps. Although we choose $f(x, \theta)$ as the laptime, explicit entropic terms can also be included to further encourage diversity within a single vertical chain or across the population.

At the conclusion of AADAPT, we use the coldest population of D agents at inverse temperature $\beta_1(T)$ to build a DPP sampler. Specifically, define the matrix H via configurations $x^{1,\cdot}$ at the lowest temperature

$$H_{ab} = \|x^{1,a} - x^{1,b}\|. \tag{3}$$

Then we define the DPP kernel K as $K_{ab} = \exp\left(-H_{ab}^2/\sigma^2\right)$ with a scale parameter $\sigma = 0.5$, and we sample $d \leq D$ configurations from this DPP.

3. Online learning with computation budgets

Now we exploit the population of d learned prototype behaviors to enable robust performance. The agent's (our) goal is to act robustly against uncertainty in opponent behaviors and adapt online to a given opponent. We parametrize the agent's (stochastic) policy as follows. At each time step, we sample goal states (consisting of pose and velocity) via a generative model $G(\theta)$ parametrized by θ (as in Section 2). For a given goal state, we compute the parameters of a cubic spline that reaches the goal by solving a nonconvex trajectory optimization problem (McNaughton, 2011); on this proposed trajectory we evaluate a collection of cost functions (such as the maximum curvature or minimum acceleration along the path) weighted by the vector x (recall Section 2), similar to Sadat et al. (2019) (see Appendix D for a description of all costs). Finally, we choose the sampled goal trajectory with minimum robust cost and perform an action to track this trajectory.

Some of the costs that evaluate the utility of a goal state involve beliefs about the opponent's future trajectory. For a goal p, we rewrite the performance objective at time t with respect to a protoype opponent i as a receding-horizon cost

$$c_i(t;p) := -\sum_{s>t} \lambda^{s-t}\mathbb{E}[r(o(s);p)],$$

where we omit dependence on the agent's cost weights x for convenience. We parametrize the agent's belief of the opponent's behavior as a categorical distribution of beliefs over the prototypes. Specifically, let $w(t) \in \Delta$ be a weight vector at a given time t, where $\Delta := \{a \in \mathbb{R}^d_+ \mid a^T\mathbf{1} = 1\}$, and let $P_0(t) := \text{Categorical}(w(t))$. Then $P_0(t)$ is the nominal distribution describing the agent's belief about the opponent. Furthermore, we consider ambiguity sets $\mathcal{P}(t)$ defined by divergence measures on the space of probability measures over Δ. For a convex function ϕ with $\phi(1) = 0$, the ϕ-divergence between distributions P and Q is $D_\phi\left(P\|Q\right) = \int \phi(\frac{dP}{dQ})dQ$. We use sets $\mathcal{P}(t) := \{Q : D_\phi\left(Q\|P_0\right)(t) \leq \rho\}$ where $\rho > 0$ is a specified constant. Our implementation employs the χ^2-divergence $\phi(t) = t^2 - 1$.

Having defined the ambiguity set $\mathcal{P}(t)$ and the cost with respect to each prototype opponent, we rewrite the robust performance objective (1) to clearly illustrate the optimization problem. Let $C(t;p)$ be a random variable representing the expected cost with respect to the belief of the opponent (and goal state p). Then the robust cost at time t is

$$\sup_{Q\in\mathcal{P}(t)} \mathbb{E}_Q[C(t;p)] = \sup_{q:\sum_i w_i\phi(\frac{q_i}{w_i})\leq\rho} \sum_i q_i c_i(t;p). \quad (4)$$

When $\rho = 0$, this is the expected cost under P_0; larger ρ adds robustness. Solving the convex optimization problem (4) first requires computing the costs $c_i(t)$. Using $\lambda \geq 0$ for the constraint $D_\phi\left(Q\|P_0\right) \leq \rho$, a partial Lagrangian is

$$\mathcal{L}(q,\lambda) = \sum_i q_i c_i(t) - \lambda\left(\sum_i w_i\phi\left(q_i/w_i\right) - \rho\right).$$

The corresponding dual function is $v(\lambda) = \sup_{q\in\Delta}\mathcal{L}(q,\lambda)$, and minimizing $v(\lambda)$ via bisection yields the solution to problem (4). Maximizing $\mathcal{L}(q,\lambda)$ with respect to q for a given λ requires $O(d)$ time using a variant of median-based search (Duchi et al., 2008) (see Appendix B). Thus, computing an ϵ-suboptimal solution uses $O(d\log(1/\epsilon))$ time.

The supremum in the robust cost (4) is over belief ambiguity. Thus, our approach generalizes beyond the goal-sampling and trajectory-optimization approach presented at the beginning of this section; it is compatible with any policy that minimizes a cost $c_i(t)$ with respect to a parametrization for opponent i's policy. In this way, it is straightforward to combine our framework with robust model predictive control formulations that have rigorous stability guarantees.

In order to perform competitive actions, the agent updates the ambiguity set $\mathcal{P}(t)$ and computes the robust cost (4) on an embedded processor on board the vehicle in real-time (*e.g.* within 100 milliseconds). In the next two subsections, we describe how to perform both operations in the presence of a severely limited computational budget, and we quantitatively analyze the implications of the budget on the robustness/performance tradeoff.

3.1. Approximating the robust cost

For a large library of prototypical opponents (large d), computing every c_i in the objective (4) is prohibitively expensive. Instead, we consider an empirical approximation of the objective, where we draw N_w indices $J_k \overset{\text{i.i.d.}}{\sim} P_0(t)$ (where $N_w < d$) and consider the weighted sum of these costs c_{j_k}. Specifically, we define the empirical approximation $\mathcal{P}_{N_w} := \{q : D_\phi\left(q\|\mathbf{1}/N_w\right) \leq \rho\}$ to $\mathcal{P}$ and solve the following empirical version of problem (4):

$$\underset{q\in\mathcal{P}_{N_w}}{\text{maximize}} \sum_k q_k c_{j_k}(t;p). \quad (5)$$

This optimization problem (5) makes manifest the price of robustness in two ways. The first involves the setup of the problem—computing the c_{j_k}. First, we denote the empirical distribution as $\hat{w}(t)$ with $\hat{w}_i(t) = \sum_k^{N_w} \mathbf{1}\{j_k = i\}/N_w$. Even for relatively small N_w/d, $\hat{w}(t)$ concentrates closely around $w(t)$ (see *e.g.* Weissman et al. (2003) for a high-probability bound). Thus, when the vehicle's belief about its opponent $w(t)$ is nearly uniform, the j_k values have few repeats. Conversely, when the belief is peaked at a few opponents, the number of unique indices is much smaller than N_w, allowing faster computation of c_{j_k}. The short setup-time enables faster planning or, alternatively, the ability to compute the costs c_{j_k} with longer horizons. Therefore, theoretical performance automatically improves as the vehicle learns about the opponent and the robust evaluation approaches the true cost.

The second way we illustrate the price of robustness is by quantifying the quality of the approximation (5) with respect to the number of samples N_w. For shorthand, define the true expected and approximate expected costs for goal p and distributions Q and q respectively as

$$R(Q;p) := \mathbb{E}_Q[C(t;p)], \quad \hat{R}(q;p) := \frac{1}{N_w}\sum_{k=1}^{N_w} q_k c_{j_k}(t;p).$$

Then, we have the following bound:

Proposition 1 (Approximation quality)**.** *Suppose $C(t;p) \in [-1,1]$ for all t, p. Let $A_\rho = \frac{2(\rho+1)}{\sqrt{1+\rho}-1}$ and $B_\rho = \sqrt{8(1+\rho)}$. Then with probability at least $1-\delta$ over the N_w samples $J_k \overset{\text{i.i.d.}}{\sim} P_0$,*

$$\left| \sup_{q\in\mathcal{P}_{N_w}} \hat{R}(q;p) - \sup_{Q\in\mathcal{P}} R(Q;p) \right| \le 4A_\rho\sqrt{\frac{\log(2N_w)}{N_w}} + B_\rho\sqrt{\frac{\log\frac{2}{\delta}}{N_w}}$$

See Appendix B for the proof. Intuitively, increasing accuracy of the robust cost requires more samples (larger N_w), which comes at the expense of computation time. Similar to computing the full cost (4), ϵ-optimal solutions require $O(N_u \log(1/\epsilon))$ time for $N_u \le N_w$ unique indices j_k. In our experiments (*cf.* Section 4), most of the computation time involves the setup to compute the N_u costs c_{j_k}.

3.2. Updating the ambiguity set

To maximize performance against an opponent, the agent updates the ambiguity set $\mathcal{P}$ as the race progresses. Since we consider ϕ-divergence balls of fixed size ρ, this update involves only the nominal belief vector $w(t)$. As with computation of the robust cost, this update must occur efficiently due to time and computational constraints.

For a given sequence of observations of the opponent $o^H_{\text{opp}}(t) := \{o_{\text{opp}}(t), o_{\text{opp}}(t-1), ..., o_{\text{opp}}(t-h+1)\}$ over a horizon h, we define the likelihood of this sequence coming from the i^{th} prototype opponent as

$$l^h_i(t) = \log d\mathbb{P}\left(o^h_{\text{opp}}(t) | G(\theta^{1,i})\right), \tag{6}$$

where $G(\theta^{1,i})$ is a generative model of goal states for the i^{th} prototype opponent. Letting $\bar{l}$ be a uniform upper bound on $l^h_i(t)$, we define the losses $L_i(t) := 1 - l^h_i(t)/\bar{l}$.

If we had enough time/computation budget, we could compute $L_i(t)$ for all prototype opponents i and perform an online mirror descent update with an entropic Bregman divergence (Shalev-Shwartz et al., 2012). In a resource-constrained setting, we can only select a few of these losses, so we use EXP3 (Auer et al., 2002) to update $w(t)$. Unlike a standard adversarial bandit setting, where we pull just one arm (*e.g.*compute a loss $L_i(t)$) at every time step, we may have resources to compute up to N_w losses in parallel at any given time (the same indices J_k discussed in Section 3.1). Denote our unbiased subgradient estimate as $\gamma(t)$:

$$\gamma_i(t) = \frac{1}{N_w}\sum_{k=1}^{N_w} \frac{L_i(t)}{w_i(t)} \mathbf{1}\{J_k = i\}. \tag{7}$$

Algorithm 2 EXP3 with N_w arm-pulls per iteration

```
Input: Stepsize sequence $\eta_t$, $w(0) := \mathbf{1}/d$, steps $T$
for $t = 0$ to $T-1$
    Sample $N_w$ indices $J_k \overset{\text{i.i.d.}}{\sim}$ Categorical($w(t)$)
    Compute $\gamma(t)$ (Equation (7))
    $w_i(t+1) := \frac{w_i(t)\exp(-\eta_t\gamma_i(t))}{\sum_{j=1}^d w_j(t)\exp(-\eta_t\gamma_j(t))}$
```

Algorithm 2 describes our slightly modified EXP3 algorithm, which has the following expected regret.

Proposition 2. *Let $z := \frac{d-1}{N_w} + 1$. Algorithm 2 run for T iterations with stepsize $\eta = \sqrt{\frac{2\log(d)}{zT}}$ has expected regret bounded by $\sum_{t=1}^T \mathbb{E}\left[\gamma(t)^T(w(t) - w^\star)\right] \le \sqrt{2zT\log(d)}$.*

See Appendix B for the proof. This regret bound looks similar to that if we simply ran N_w standard EXP3 steps per iteration t (in which case $z = d/N_w$). However, our approach enables parallel computation which is critical in our time-constrained setting. Note that the "multiple-play" setting we propose here has been studied before with better regret bounds but higher computational complexity per iteration (Uchiya et al., 2010; Zhou & Tomlin, 2018). We prefer our approach for its simplicity and ability to be easily combined with the robust-cost computation.

4. Experiments

In this section we first describe the AR environment used to conduct our experiments. Next we explore the hyperparameters of the algorithms in Section 2 and 3, identifying a preferred configuration. Then we consider the overarching hypothesis: online adaptation can improve the performance of robust control strategies. We show the statistically significant results affirming the theory and validate the approach's performance on real vehicles.

The experiments use an existing low-cost 1/10th-scale, Ackermann-steered AV (Figure 2). Additionally, we create a simulator and an associated OpenAI Gym API (Brockman et al., 2016) suitable for distributed computing. The simulator supports multiple agents as well as deterministic executions. We experimentally determine the physical parameters of the agent models for simulation and use SLAM to build the virtual track as a mirror of a real location (see Figure 4). The hardware specifications, software, and simulator are open-source [2] (see Appendices C and D for details).

The agent software uses a hierarchical planner (Gat et al., 1998) similar to Ferguson et al. (2008). The key difference is the use of a masked autoregressive flow (MAF) (Rezende & Mohamed, 2015) which provides the generative model for

[2] https://github.com/travelbureau/f0_icml_code

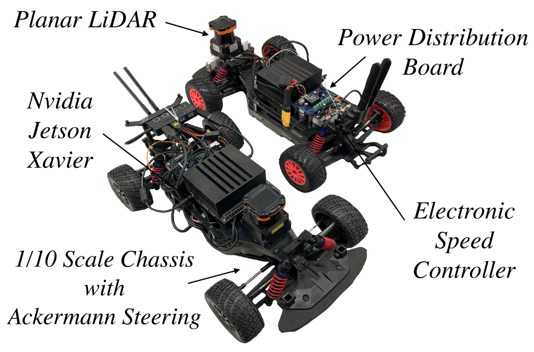
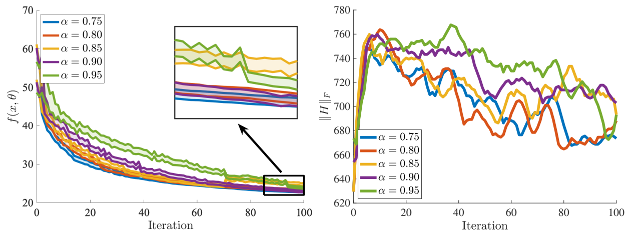

Planar LiDAR
Power Distribution Board
Nvidia Jetson Xavier
Electronic Speed Controller
1/10 Scale Chassis with Ackermann Steering

Figure 2. Components of the 1/10-scale vehicle



(a) Performance vs. iteration (b) Diversity vs. iteration

Figure 3. Hyperparameter selection for AADAPT. (a) 95%-confidence intervals for $f(x, \theta)$ in the coldest temperature level. (b) Frobenius norm of the Mahalanobis distance matrix H (3). The value $\alpha = 0.9$ achieves the best performance and diversity.

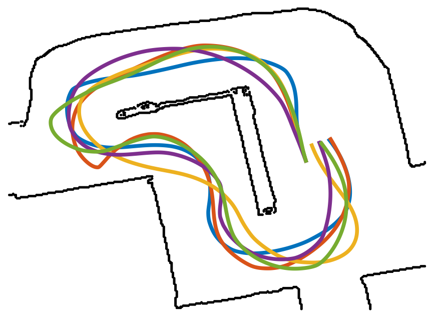
goal states, $G(\theta)$. Belief inference and robust cost computation require sampling and evaluating the likelihood of goal states. MAFs can evaluate likelihoods quickly but generate samples slowly. Inspired by Oord et al. (2018) we overcome this inefficency by training a "student" inverse autogressive flow (IAF) (Kingma et al., 2016) on MAF samples. Given a sample of goals from the IAF, the agent synthesizes dynamically feasible trajectories following McNaughton (2011). Each sample is evaluated according to Equation 4; the weights of the cost functions are learned by AADAPT (and formal definitions of the cost components are in Appendix D). Belief updates use Algorithm 2 using the MAF to compute the losses $L_i(t)$.

4.1. Offline population synthesis

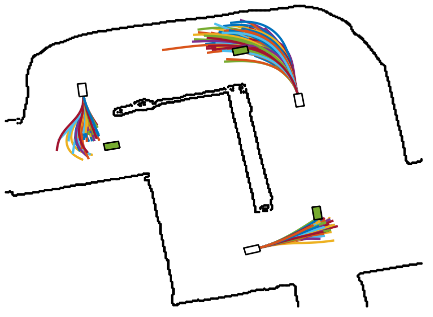
We run AADAPT with $L = 5$ populations, $D = 160$ configurations per population, and $T = 100$ iterations. For vertical MCMC steps, we randomly sample 16 configuratons per population and perform $V = 2$ iterations of 5 hit-and-run proposals. Furthermore, we perform $E = DL^2/\alpha^{t/(L-1)}$ horizontal steps (motivated by the fact fact that "tunneling" from the highest-temperature level to the coldest takes $O(L^2)$ accepted steps). Finally, for training θ, we use Adam (Kingma & Ba, 2014) with a learning rate of 10^{-4}.

Figure 3 shows results with 5 choices for the most influential hyperparameter, the annealing rate: $\alpha \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$. Figure 3(a) displays 95%-confidence intervals for the mean laptime in the coldest level. The annealing rates $\alpha \in \{0.75, 0.80, 0.90\}$ all result in comparable performance of 22.95 ± 0.14 (mean $\pm$ standard error) seconds at the end of the two-lap run. Figure 3(b) illustrates a metric for measuring diversity, the Frobenius norm of the Mahalanobis distance matrix (3). We see that $\alpha = 0.9$ results in the highest diversity while also attaining the best performance. Thus, in further experimentation, we use the results from the run conducted with $\alpha = 0.9$.

Figure 4 illustrates qualitative differences between cost functions. Figure 4(a) displays trajectories for agents driven using 5 cost functions sampled from the learned DPP. The



(a) Rollouts from 5 agents



(b) Snapshot trajectories

Figure 4. Qualitative illustrations of multimodal behavior in the learned population of cost functions

cornering behavior is quite different between the trajectories. Figure 4(b) displays the trajectories chosen by all 160 agents in the population at $\beta_1(T)$ at various snapshots along the track. There is a wider spread of behavior near turns than areas where the car simply drives straight.

4.2. Simulated experiments

We conduct a series of tests in simulation to determine the effects of distributional robustness and adaptivity on overall safety and performance. For a given robustness level $\rho/N_w \in \{0.001, 0.025, 0.2, 0.4, 0.75, 1.0\}$ (with $N_w = 8$ for all experiments), we simulate 40 two-lap races against each of the $d = 10$ diverse opponents sampled from the DPP. For fair comparisons, half of the races have the opponent starting on the outside and the other half with the opponent on the inside of the track. Importantly, these experiments in-

Table 1. The effect of distributional robustness on aggressiveness

Agent	% of iTTC values < 0.5s
$\rho/N_w = 0.001$	7.86± 0.90
$\rho/N_w = 0.025$	6.46± 0.78
$\rho/N_w = 0.2$	4.75± 0.65
$\rho/N_w = 0.4$	5.41± 0.74
$\rho/N_w = 0.75$	5.50± 0.82
$\rho/N_w = 1.0$	5.76± 0.84

Table 2. The effect of adaptivity on win-rate

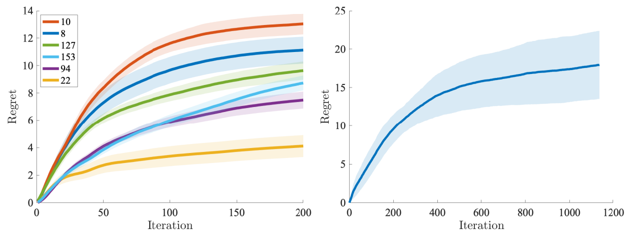
Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.593± 0.025	0.588± 0.025	0.84
$\rho/N_w = 0.025$	0.593± 0.025	0.600± 0.024	0.77
$\rho/N_w = 0.2$	0.538± 0.025	0.588± 0.025	0.045
$\rho/N_w = 0.4$	0.503± 0.025	0.573± 0.025	0.0098
$\rho/N_w = 0.75$	0.513± 0.025	0.593± 0.025	0.0013
$\rho/N_w = 1.0$	0.498± 0.025	0.590± 0.025	0.00024

volve only the most elite policies from the temperature level $\beta_1(T)$. Since the physical characteristics of the vehicles are identical, win rates between elite policies significantly greater than 0.5 are meaningful. In contrast, against a set of weaker opponents sampled via DPP from the 3rd temperature level $\beta_3(T)$, the win-rate (fraction of races that our agent from the coldest temperature wins) is 0.848 ± 0.012.

Effects of distributional robustness We test the hypothesis that distributional robustness results in more conservative policies. For every race both agents have a fixed robustness level ρ and no adaptivity. To measure aggressiveness/conservativeness, we consider instantaneous time-to-collision (iTTC) of the vehicles during the race (see Appendix F). Smaller iTTC values imply more dangerous scenarios and more aggressive policies. In Table 1, we track the rate at which iTTC < 0.5 seconds. As expected, aggressiveness decreases with robustness (the rate of small iTTC values decreases as ρ increases). The trend is $a + b\log(\rho)$, where $a = 5.16 \pm 0.34$ and $b = -0.36 \pm 0.10$ ($R^2 = 0.75$).

Effects of adaptivity Now we investigate the effects of online learning on the outcomes of races. Figure 5(a) shows that Algorithm 2 identifies the opponent vehicle within approximately 150 timesteps (15 seconds), as illustrated by the settling of the regret curve.[3] Given evidence that the opponent model can be identified, we investigate whether adaptivity improves performance, as measured by win-rate. Table 2 displays results of paired t-tests for multiple robustness levels (with a null-hypothesis that adaptivity does not change the win-rate). Each test compares the effect of adaptivity for our agent on the 400 paired trials (and the opponents are always nonadaptive). Adaptivity significantly improves performance for the larger robustness levels $\rho/N_w \geq 0.2$.

[3]We omit 3 of the regret lines for clarity in the plot.



(a) Simulation (b) Real

Figure 5. 95%-confidence intervals for regret using $N_w = 8$ arms in (a) simulation and (b) reality. The legend in (a) denotes opponent id and the opponent in (b) has id 22. Our agent has id 33.

As hypothesized above, adaptivity automatically increases aggressiveness as the agent learns about its opponent and samples fewer of the other arms to compute the empirical robust cost (5). This effect is more prominent when robustness levels are greater, where adaptivity brings the win-rate back to its level without robustness ($\rho/N_w = 0.001$). Thus, the agent successfully balances safety and performance by combining distributional robustness with adaptivity.

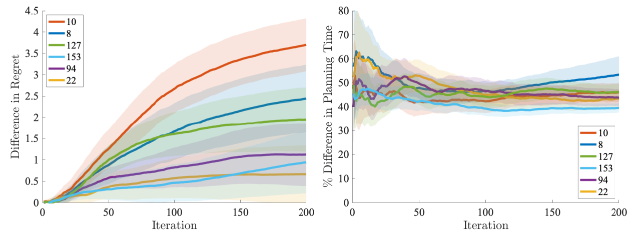
4.3. Real-world validation

The real world experiments consist of races between agents 22 and 33; we examine the transfer of the opponent modeling approach from simulation to reality. In Figure 5(b) we plot 33's cumulative regret; it takes roughly 4 times as many observations relative to simulation-based experiments to identify the opponent (agent 22). We demonstrate the qualitative properties of the experiments in a video of real rollouts synchronized with corresponding simulations.[4] State estimation error and measurement noise drive the gap between simulated and real performance. First, both vehicle poses are estimated with a particle filter, whereas simulation uses ground-truth states. Since we infer beliefs about an opponent's policy based on a prediction of their actions at a given state, pose estimation error negatively impacts the accuracy of this inference. Second, the simulator only captures the geometry of the track; in reality glass and metal surfaces significantly affect the LIDAR range measurements, which in turn impact the MAF and IAF networks. The convergence of the cumulative regret in Figure 5(b) reflects that, despite the simulation/reality gap, our simulation-trained approach transfers to the real world. Diminishing the effect of the simulation/reality gap is the subject of future work (see Appendix E).

4.4. Approximation analysis

Sampling N_w indices $J_k \overset{\text{i.i.d.}}{\sim} P_0(t)$ allows us to quickly compute the approximate robust cost (Section 3.1) and perform a bandit-style update to the ambiguity set (Section

[4]https://youtu.be/7Yat9FZzE4g



(a) Difference in regret (b) Difference in planning time

Figure 6. 95%-confidence intervals for the (a) difference in regret and (b) percent difference in cumulative planning time when using sampling approximations vs. online mirror descent. Online mirror descent yields lower regret at the expense of longer planning times.

3.2). Now we analyze the time-accuracy tradeoff of performing this sampling approximation rather than using all d prototypical opponents at every time step. Figure 6(a) shows the difference in regret for the same experiments as in Figure 5(a) if we perform full online mirror-descent updates. Denoting the simulations in Figure 5(a) as S and those with the full mirror descent update as M, we compute difference as $\text{Regret}_S - \text{Regret}_M$. As expected, the difference is positive, since receiving the true gradient is better than the noisy estimate (7). Similarly, Figure 6(b) shows the percent increase in cumulative planning time for the same pairs (sampling vs. full online mirror descent), where percent increase is given by $100(\text{Time}_M - \text{Time}_S)/\text{Time}_S$. As the agent learns who the opponent is, it draws many repeats in the N_w arms, whereas the full mirror descent update always performs d computations. As a result, the percent increase in cumulative iteration time approaches a contant of approximately $1.5\times$. All of these comparisons are done in simulation, where the agent is not constrained to perform actions in under 100 milliseconds. Performing a full mirror descent update is impossible on the real car, as it requires too much time.

4.5. Out-of-distribution opponents

Now we measure performance against two agents—OOD1 and OOD2—that are not in the distribution developed by our offline population synthesis approach (see Appendix F.2 for details on each agent's policy). We perform only simulated experiments, as we are unable to perform further real-world experimentation at the time of writing due to the COVID-19 pandemic. For given robustness levels $\rho/N_w \in \{0.001, 1.0\}$ and $N_w = 8$ for all experiments, we perform 180 two-lap races against each of the two human-created racing agents. Again, for fair comparison, half of the experiments have the opponent start on the outside and half on the inside. Tables 3 and 4 show the results. Overall, the trends match those of the in-distribution opponents. Namely, adaptivity significantly increases the win-rate when robustness is high ($\rho/N_w = 1.0$), whereas for low robustness ($\rho/N_w = 0.001$) there is no significant change. Interestingly, adaptivity with robustness not only recovers but surpasses the win-rate of the non-adaptive non-robust policy. We hypothesize that, because out-of-distribution opponents do not match any of the learned prototypes, maintaining an uncertainty over belief automatically helps the agent plan against the "surprising" out-of-distribution actions. Validation of this hypothesis by comparing performance against more out-of-distribution opponents is an interesting direction for future work. Overall, we observe that even against out-of-distribution opponents, we achieve the overall goal of balancing performance and safety.

Table 3. The effect of adaptivity on win-rate vs. OOD1

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.633±0.036	0.683±0.035	0.280
$\rho/N_w = 1.0$	0.483±0.037	0.717±0.034	5.721E-6

Table 4. The effect of adaptivity on win-rate vs. OOD2

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.494±0.037	0.589±0.037	0.059
$\rho/N_w = 1.0$	0.572±0.037	0.739±0.033	0.001

5. Conclusion

The central hypothesis of this paper is that distributionally robust evaluation of plans relative to the agent's belief state about opponents, which is updated as new observations are made, can lead to policies achieving the same performance as non-robust approaches without sacrificing safety. To evaluate this hypothesis we identify a natural division of the underlying problem. First, we parameterize the set of possible opponents via population-based synthesis without requiring expert demonstrations. Second, we propose an online opponent-modeling framework which enables the application of distributionally robust optimization (DRO) techniques under computational constraints. We provide strong empirical evidence that distributional robustness combined with adaptivity enables a principled method automatically trading between safety and performance. Also, we demonstrate the transfer of our methods from simulation to real autonomous racecars. The addition of recursive feasibility arguments for stronger safety guarantees could improve the applicability of these techniques to real-world settings. Furthermore, although autonomous racing is the current focus of our experiments, future work should explore the generality of our approach in other settings such as human-robot interaction.

Acknowledgements

AS and JD were partially supported by the DAWN Consortium, NSF CAREER CCF-1553086, NSF HDR 1934578 (Stanford Data Science Collaboratory), ONR YIP N00014-19-2288, and the Sloan Foundation. RT was supported by the Department of the Navy and ONR, Award Nos. N00014-17-1-2699 and N00014-18-1-2210.

References

Abernethy, J. and Rakhlin, A. Beating the adaptive bandit with high probability. In *2009 Information Theory and Applications Workshop*, pp. 280–289. IEEE, 2009.

Agarwal, N., Bullins, B., Hazan, E., Kakade, S. M., and Singh, K. Online control with adversarial disturbances. *arXiv preprint arXiv:1902.08721*, 2019.

Althoff, M. and Dolan, J. M. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

Althoff, M., Koschi, M., and Manzinger, S. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 719–726. IEEE, 2017.

Arenz, O., Zhong, M., Neumann, G., et al. Efficient gradient-free variational inference using policy search. 2018.

Arulkumaran, K., Cully, A., and Togelius, J. Alphastar: An evolutionary computation perspective. *arXiv preprint arXiv:1902.01724*, 2019.

Åström, K. J. and Wittenmark, B. *Adaptive control*. Courier Corporation, 2013.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Bagnell, J. A., Ng, A. Y., and Schneider, J. G. Solving uncertain markov decision processes. 2001.

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

Bélisle, C. J., Romeijn, H. E., and Smith, R. L. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266, 1993.

Bemporad, A. and Morari, M. Robust model predictive control: A survey. In *Robustness in identification and control*, pp. 207–226. Springer, 1999.

Ben-Tal, A., den Hertog, D., Waegenaere, A. D., Melenberg, B., and Rennen, G. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dkebiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bertsimas, D. and Sim, M. The price of robustness. *Operations research*, 52(1):35–53, 2004.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Bubeck, S., Cesa-Bianchi, N., et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5 (1):1–122, 2012.

Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.

Černỳ, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

Coulter, R. C. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

Dean, J. and Ghemawat, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pp. 4188–4197, 2018.

Ding, W. and Shen, S. Online vehicle trajectory prediction using policy anticipation network and optimization-based context reasoning. *arXiv preprint arXiv:1903.00847*, 2019.

Doyle, J., Glover, K., Khargonekar, P., and Francis, B. State-space solutions to standard h_2 and h_∞ control problems. In *1988 American Control Conference*, pp. 1691–1696. IEEE, 1988.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto thel1-ball for learning in high dimensions. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008. doi: 10.1145/1390156.1390191. URL http://dx.doi.org/10.1145/1390156.1390191.

Federation Internationale de l'Automobile. Formula one 2019 results. https://www.formula1.com/en/results.html/2019/, 2019.

Ferguson, D., Howard, T. M., and Likhachev, M. Motion planning in urban environments. *Journal of Field Robotics*, 25(11-12):939–960, 2008.

Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pp. 9516–9527, 2018.

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Galceran, E., Cunningham, A. G., Eustice, R. M., and Olson, E. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction. In *Robotics: Science and Systems*, volume 1, 2015.

Gao, Y., Gray, A., Tseng, H. E., and Borrelli, F. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014.

Gat, E., Bonnasso, R. P., Murphy, R., et al. On three-layer architectures. *Artificial intelligence and mobile robots*, 195:210, 1998.

Geyer, C. J. Markov chain monte carlo maximum likelihood. 1991.

Gilboa, I. and Marinacci, M. Ambiguity and the bayesian paradigm. In *Readings in formal epistemology*, pp. 385–439. Springer, 2016.

Gleave, A., Dennis, M., Kant, N., Wild, C., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

Hastings, W. K. Monte carlo sampling methods using markov chains and their applications. 1970.

He, H., Boyd-Graber, J., Kwok, K., and Daumé III, H. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pp. 1804–1813, 2016.

Hess, W., Kohler, D., Rapp, H., and Andor, D. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278. IEEE, 2016.

Hintjens, P. *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

Howard, T. M. *Adaptive model-predictive motion planning for navigation in complex environments*. Carnegie Mellon University, 2009.

Hu, J. and Hu, P. Annealing adaptive search, cross-entropy, and stochastic approximation in global optimization. *Naval Research Logistics (NRL)*, 58(5):457–477, 2011.

Ingber, L. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Karaman, S. and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Kelly, A. and Nagy, B. Reactive nonholonomic trajectory generation via parametric optimal control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Kochenderfer, M. J. *Decision making under uncertainty: theory and application*. MIT press, 2015.

Kulesza, A., Taskar, B., et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Kumar, P. R. A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23 (3):329–380, 1985.

Liniger, A. and Lygeros, J. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 2019.

Lovász, L. Hit-and-run mixes fast. *Mathematical Programming*, 86(3):443–461, 1999.

Lovász, L. and Vempala, S. Hit-and-run is fast and fun. *preprint, Microsoft Research*, 2003.

Lovász, L. and Vempala, S. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

Luenberger, D. *Optimization by Vector Space Methods*. Wiley, 1969.

Majumdar, A. and Tedrake, R. Robust online motion planning with regions of finite time invariance. In *Algorithmic foundations of robotics X*, pp. 543–558. Springer, 2013.

Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., and Savarese, S. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3932–3939. IEEE, 2017.

Marinari, E. and Parisi, G. Simulated tempering: a new monte carlo scheme. *EPL (Europhysics Letters)*, 19(6): 451, 1992.

Matyas, J. Random optimization. *Automation and Remote control*, 26(2):246–253, 1965.

McNaughton, M. Parallel algorithms for real-time motion planning. 2011.

Mouret, J.-B. and Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Nagy, B. and Kelly, A. Trajectory generation for car-like robots using cubic curvature polynomials. *Field and Service Robots*, 11, 2001.

Namkoong, H. and Duchi, J. C. Variance regularization with convex objectives. In *Advances in Neural Information Processing Systems 30*, 2017.

Nilim, A. and El Ghaoui, L. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Norden, J., O'Kelly, M., and Sinha, A. Efficient black-box assessment of autonomous vehicle safety. *arXiv preprint arXiv:1912.03618*, 2019.

O'Kelly, M., Zheng, H., Auckley, J., Jain, A., Luong, K., and Mangharam, R. Technical Report: TunerCar: A Superoptimization Toolchain for Autonomous Racing. Technical Report UPenn-ESE-09-15, University of Pennsylvania, September 2019. https://repository.upenn.edu/mlab_papers/122/.

Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, pp. 3918–3926, 2018.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2817–2826. JMLR. org, 2017.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pp. 1530–1538. JMLR. org, 2015.

Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.-C., Yumer, E., and Urtasun, R. Jointly learnable behavior and trajectory planning for self-driving vehicles. *arXiv preprint arXiv:1910.04586*, 2019.

Sadigh, D., Sastry, S., Seshia, S. A., and Dragan, A. D. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, volume 2. Ann Arbor, MI, USA, 2016.

Samson, P. Concentration of measure inequalities for Markov chains and ϕ-mixing processes. *Annals of Probability*, 28(1):416–461, 2000.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

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

Sinha, A. and Duchi, J. C. Learning kernels with random features. In *Advances in Neural Information Processing Systems*, pp. 1298–1306, 2016.

Sinha, A., Namkoong, H., and Duchi, J. Certifiable distributional robustness with principled adversarial training. In *Proceedings of the Fifth International Conference on Learning Representations*, 2017. arXiv:1710.10571 [cs.LG].

Smirnova, E., Dohmatob, E., and Mary, J. Distributionally robust reinforcement learning. *arXiv preprint arXiv:1902.08708*, 2019.

Smith, R. L. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

Snider, J. M. et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

Sontges, S., Koschi, M., and Althoff, M. Worst-case analysis of the time-to-react using reachable sets. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1891–1897. IEEE, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Swendsen, R. H. and Wang, J.-S. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57 (21):2607, 1986.

Tamar, A., Mannor, S., and Xu, H. Scaling up robust mdps using function approximation. In *International Conference on Machine Learning*, pp. 181–189, 2014.

Uchiya, T., Nakamura, A., and Kudo, M. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory*, pp. 375–389. Springer, 2010.

Van Den Berg, J., Abbeel, P., and Goldberg, K. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

Vedder, B. Vedder electronic speed controller. URL `https://vesc-project.com/documentation`.

Vinnicombe, G. Frequency domain uncertainty and the graph topology. *IEEE Transactions on Automatic Control*, 38(9):1371–1383, 1993.

Walsh, C. and Karaman, S. Cddt: Fast approximate 2d ray casting for accelerated localization. abs/1705.01167, 2017. URL `http://arxiv.org/abs/1705.01167`.

Wang, Z., Spica, R., and Schwager, M. Game theoretic motion planning for multi-robot racing. In *Distributed Autonomous Robotic Systems*, pp. 225–238. Springer, 2019.

Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., and Weinberger, M. J. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

Williams, G., Goldfain, B., Drews, P., Rehg, J. M., and Theodorou, E. A. Autonomous racing with autorally vehicles and differential games. *arXiv preprint arXiv:1707.04540*, 2017.

Zhou, D. P. and Tomlin, C. J. Budget-constrained multi-armed bandits with multiple plays. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Zhou, K., Doyle, J. C., and Glover, K. Robust and optimal control. 1996.