Performance-driven Constrained Optimal Auto-Tuner for MPC

Albert Gassol Puigjaner, Manish Prajapat, Andrea Carron, Andreas Krause, Melanie N. Zeilinger

***Abstract*— A key challenge in tuning Model Predictive Control (MPC) cost function parameters is to ensure that the system performance stays consistently above a certain threshold. To address this challenge, we propose a novel method, COAT-MPC, Constrained Optimal Auto-Tuner for MPC. With every tuning iteration, COAT-MPC gathers performance data and learns by updating its posterior belief. It explores the tuning parameters' domain towards optimistic parameters in a goal-directed fashion, which is key to its sample efficiency. We theoretically analyze COAT-MPC, showing that it satisfies performance constraints with arbitrarily high probability at all times and provably converges to the optimum performance within a finite time. Through comprehensive simulations and comparative analyses with a hardware platform, we demonstrate the effectiveness of COAT-MPC in comparison to classical Bayesian Optimization (BO) and other state-of-the-art methods. When applied to autonomous racing, our approach outperforms baselines in terms of constraint violations and cumulative regret over time.**

COAT-MPC Code: https://github.com/albertgassol1/coat_mpc

CRS Code: https://gitlab.ethz.ch/ics/crs

Video: https://youtu.be/Ep_BX3BDaeU?si=ShPcvWB_I8xCGg9T

I. Introduction

Model Predictive Control (MPC) is a prominent optimization-based control framework that can handle constraints and optimize system performance by predicting the system's future behavior. MPC is widely used in many robotic applications such as autonomous driving [1], four-legged robots [2], and bipedal robots [3]. While MPC is a successful optimal control technique, one of the significant challenges in its implementation is tuning the cost function parameters. Mainly, designing a cost function that balances competing objectives is a non-trivial task that requires significant trial and error. Moreover, the cost function parameters often depend on the specific environment and system dynamics, making it difficult to design a single set of parameters that can perform well in all scenarios. Usually, the task of fine-tuning cost function parameters involves heuristic methods and demands expert knowledge, leading to a significant number of costly and time-consuming experimental iterations.

In most applications, we tune to maximize some performance function, e.g., while tuning for racing, we optimize the lap time. However, these performance functions are often *a-priori* unknown and need to be learned through data. Naively, to find the optimal parameters, one may try out a large set of parameters in a grid search approach. Apart from inefficiency caused by executing a large number of parameters, many of those parameters may lead the system to halt, i.e., low performance, resulting in a wasteful evaluation. For example, in tuning an MPC for autonomous racing, it is undesirable to use parameters that make the car move extremely slow, or even stop before finishing a lap. This motivates a tuning process ensuring system performance above a threshold.

All authors are with ETH Zurich.

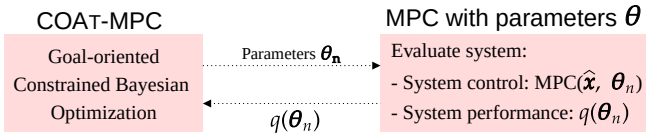


Fig. 1: COAT-MPC algorithm overview. COAT-MPC proposes a new set of cost function weights θ_n, which are evaluated on the system. The algorithm captures a noisy performance function sample, which is used to update the Bayesian Optimization surrogate model posterior and acquire a new set of cost function weights. The process is repeated until convergence.

To tackle these challenges, we propose a novel algorithm: COAT-MPC, Constrained Optimal Auto-Tuner for MPC. COAT-MPC explores the space of MPC cost function parameters and builds a belief about the *a-priori* unknown performance function through data, utilizing tools from Gaussian processes [4]. COAT-MPC incorporates safe exploration ideas from [5], [6], [7] and recursively recommends sufficiently informative parameters that ensure exploration while satisfying the performance constraint. We establish convergence guarantees to the optimal tuning parameters in a finite number of samples while ensuring performance constraint satisfaction with an arbitrarily high probability. For finite time convergence, we present a sample complexity bound by extending the analysis of [7] from continuous to discrete domains. In particular, our sample complexity result removes an explicit dependence on the discretization step size and thus significantly improves prior safe exploration results in discrete domains [6], [8], [9].

Finally, we demonstrate the effectiveness of COAT-MPC in the application of autonomous racing. We tune a Model Predictive Contouring Control [10] formulation with the objective of optimizing the lap time while avoiding undesirable effects such as halting. Our evaluation encompasses a comprehensive analysis in both, simulation and in experiments on a 1:28 scale RC racecar [11]. We present a comparative analysis against other automatic tuning methods including classical Bayesian optimization. The results demonstrate that our approach outperforms other methods in terms of the number of constraint violations and yields an improved cumulative regret over time. To the best of our knowledge, this is the first paper to present an algorithm for MPC tuning that offers optimality guarantees while satisfying a performance constraint.

II. Related works

Controller tuning in robotics has been an active area of research. Recently, data-driven methods aimed at learning the relationship between system parameters and a desired metric have emerged as promising solutions for automatic tuning. For instance, methods such as the Metropolis-Hastings algorithm [12] and Policy Search methods [13], have demonstrated state-of-the-art results in model-based agile flight control.

Bayesian Optimization (BO) [14], [15] has been particularly successful due to its ability to optimize the objective function using a limited number of samples [16]. It utilizes a probabilistic model to represent the unknown objective function, which is updated as new data is acquired. Even though any probabilistic model can be used, a popular choice to model unknown functions is a Gaussian Process (GP) [4], e.g., in BO [8], [17], [18], [19], [6], MPC [7] or experiment design [20]. In the context of MPC parameter tuning, BO can be used to find the optimal cost function parameters for a given platform and environment [21] by selecting an adequate objective function, i.e. laptime in the case of MPC tuning for racing applications. Additionally, subsequent works have introduced contextual information from the environment or system [17], considered the confidence of the probabilistic model to enhance the cautiousness and convergence rate of BO [18], and combined BO with trust region optimization [19]. However, it is worth noting that these BO methods do not take into account constraints on the objective function, which can lead to parameters that produce very poor performance due to the unbounded exploration of the algorithms. This is particularly problematic in the context of robotics, where safety and performance during testing are of paramount importance.

Several approaches have been developed to incorporate constraints into the BO algorithm. One such approach involves utilizing a variant of the Expected Improvement (EI) function, referred to as the Constrained Expected Improvement (EI_C) [22], [23], [24]. This approach involves modeling the constraint function with a prior distribution and incorporating a probability of violation into the acquisition function. However, none of the aforementioned methods provide a theoretical guarantee of constraint satisfaction, which may result in evaluating poor performance parameters.

In the literature of Constrained Bayesian Optimization (CBO), SAFE-OPT [8], [5], [25] is introduced as an algorithm that aims to provide high-probability guarantees of constraint satisfaction. The algorithm leverages the regularity assumption on the objective function and the Lipschitz continuity to identify a set of parameters where the constraints on the underlying objective function are unlikely to be violated. Even though SAFE-OPT has been proven to guarantee safety, it tends to explore the complete safe parameter region, consequently leading to sample inefficiency in relation to the optimization task.

In order to tackle sample inefficiency in safe exploration, the authors of [6] propose GOOSE, a goal-oriented safe exploration algorithm for any interactive machine learning method. GOOSE leverages the regularity assumption on the constraint function to define over- and under-approximations of the safe set. A goal within the over-approximated set is defined at each iteration with the purpose of steering the recommendations of GOOSE towards the goal while ensuring safety. However, the sample complexity analysis of GOOSE is explicitly dependent on its discretization step size, leading to poor scalability.

Previous works have several limitations: they either do not incorporate constraints in the optimization [14], [21], [17], [18], [19], lack theoretical guarantees on constraint satisfaction [22], [23], [24], suffer from sample inefficiency [8], [5], [25], or provide poorly scalable sample complexity results [6]. In this paper, we present a sample-efficient algorithm that satisfies performance constraints and offers scalable theoretical guarantees for MPC cost function tuning.

III. Problem statement

We consider a non-linear dynamical system controlled using an MPC with cost function parameters $\boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$.

$$\begin{aligned} \min_{\boldsymbol{u}_{0:\text{N}}} \quad & \sum_{i=0}^{N} l(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{x}_0 = \hat{\boldsymbol{x}}(t),\, \boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i) \\ & \boldsymbol{x}_i \in \mathcal{X},\, \boldsymbol{u}_i \in \mathcal{U}, \forall i = 0, \cdots, N\,, \end{aligned} \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^{N_x}$ is the system state with dimension N_x, $\boldsymbol{u}_i \in \mathbb{R}^{N_u}$ is the control input with dimension N_u, $\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i) : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_x}$ denotes the system dynamics, $l(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\theta}) : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R}^{N_\theta} \to \mathbb{R}$ is the cost function and $\hat{\boldsymbol{x}}(t) \in \mathbb{R}^{N_x}$ is the initial system state at time t.

We define a performance function $q : D \to \mathbb{R}$, where $D \subseteq \mathbb{R}^{N_\theta}$ is a finite domain of cost function parameters, that measures the performance of a given set of tuning parameters. The function $q(\boldsymbol{\theta})$ is *a-priori* unknown and needs to be learned with data. To learn the performance function, at any iteration n, one can control the system with an MPC using any parameter $\boldsymbol{\theta}_n \in D$ and obtain a noisy observation of $q(\boldsymbol{\theta}_n)$. We examine the problem of finding the parameters that maximize q while ensuring that the performance is above a user-specified threshold $\tau \in \mathbb{R}$ (e.g. an arbitrary upper bound lap time for racing applications) in all iterations, i.e.,

$$q(\boldsymbol{\theta}_n) \geq \tau, \forall n \geq 1. \tag{2}$$

Ideally, we do not want to execute all parameters, but only those that are essential to guarantee convergence to optimal parameters while always satisfying the constraint.

We next make an initialization assumption which is crucial to start the tuning process.

Assumption 1 (Initial seed)**.** *An initial set of parameters $\mathcal{S}_0 \subseteq D$ that satisfy the performance constraint is known, i.e., $\forall \boldsymbol{\theta} \in \mathcal{S}_0, q(\boldsymbol{\theta}) \geq \tau$.*

For a suitable τ, this assumption can be satisfied by employing an MPC capable of controlling the system to obtain measurements of the performance function q. For

instance, in autonomous racing, employing parameters of an MPC (which need not be optimized) capable of leading the car to complete the lap will meet the criteria outlined in Assumption 1.

Using Assumption 1, we construct a reachable set of cost function parameters, $\mathcal{S}^{q,\epsilon}$, which contains all the parameters that can be reached starting from the initial set $\mathcal{S}_0$, while always satisfying the performance constraint up to a statistical confidence of ϵ-margin, i.e, $q(\boldsymbol{\theta}) - \epsilon \geq \tau$ (see Section IV-B for details on how to construct this set).

COAT-MPC objective. Given the noisy measurements of the performance function, the best any algorithm can guarantee is convergence to parameters $\boldsymbol{\theta}^g$ satisfying,

$$q(\boldsymbol{\theta}^g) \geq \max_{\boldsymbol{\theta} \in \mathcal{S}^{q,\epsilon}} \quad q(\boldsymbol{\theta}) - \epsilon, \tag{3}$$

in a finite number of tuning iterations while ensuring Eq. (2).

IV. BACKGROUND

In this section, we first introduce Gaussian Processes in Section IV-A, which are used to model the unknown function q, and utilize them to explain concepts of safe exploration relevant for COAT-MPC in Section IV-B.

A. Gaussian processes

The performance function q is *a-priori* unknown. Therefore, to explore the parameter space while satisfying the performance constraint, we need a mechanism that ensures that knowing about q at a certain $\boldsymbol{\theta}$ provides us with some information about the neighboring region. To this end, we make the following regularity assumptions on the performance function q.

Assumption 2. *The domain D is endowed with a positive definite kernel $k_q(\cdot,\cdot)$, and q has a bounded norm in the associated Reproducing Kernel Hilbert Space (RKHS) [26], $||q||_k \leq B_q < \infty$.*

This assumption allows us to model the performance function q using a Gaussian Process [4]. GPs are probability distributions over a class of continuous smooth functions. GPs are characterized by a mean $\mu : \mathbb{R}^{N_\theta} \to \mathbb{R}$ and a kernel function $k : \mathbb{R}^{N_\theta} \times \mathbb{R}^{N_\theta} \to \mathbb{R}$, which captures the notion of similarity between data points. Without loss of generality, we normalize such that $k(\boldsymbol{\theta}, \boldsymbol{\theta}) \leq 1, \forall \boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$. Given a set of n noisy samples collected at $A_n = \{\boldsymbol{\theta}_i\}_{i=1}^n$, perturbed by η_n conditionally σ_η-sub-Gaussian noise, given by $\boldsymbol{y}_n = [q(\boldsymbol{\theta}_1) + \eta_1, \ldots, q(\boldsymbol{\theta}_n) + \eta_n]^\top$, we can compute the posterior over q in closed form using,

$$\begin{aligned} \mu_n(\boldsymbol{\theta}) &= \boldsymbol{k}_n^\top(\boldsymbol{\theta})(\boldsymbol{K}_n + \boldsymbol{I}_n\sigma_\eta^2)^{-1}\boldsymbol{y}_n, \\ k_n(\boldsymbol{\theta}, \boldsymbol{\theta}') &= k_n(\boldsymbol{\theta}, \boldsymbol{\theta}') - \boldsymbol{k}_n^\top(\boldsymbol{\theta})(\boldsymbol{K}_n + \boldsymbol{I}_n\sigma_\eta^2)^{-1}\boldsymbol{k}_n(\boldsymbol{\theta}'), \\ \sigma_n(\boldsymbol{\theta}) &= \sqrt{k_n(\boldsymbol{\theta}, \boldsymbol{\theta})}, \end{aligned} \tag{4}$$

where the covariance matrix $\boldsymbol{K}_n$ is defined as $\boldsymbol{K}_n(i,j) = k_n(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, $i,j \in \{1,\ldots,n\}$, and $\boldsymbol{k}_n(\boldsymbol{\theta}) = [k_n(\boldsymbol{\theta}_1, \boldsymbol{\theta}), \ldots, k_n(\boldsymbol{\theta}_n, \boldsymbol{\theta})]^\top$ and $\sigma_n : \mathbb{R}^{N_\theta} \to \mathbb{R}$ denotes the predictive variance.

Assumption 2 is a typical assumption in prior works that use GPs to model unknown functions [17], [6], [8]. We consider that the performance function q is L-Lipschitz continuous with respect to some metric d on D, e.g. the Euclidean metric. This is automatically satisfied when using common isotropic kernels, such as the Mátern and Gaussian kernels. Additionally, we define the maximum *information capacity* $\gamma_n \coloneqq \sup_{A \subseteq D\,:\,|A| \leq n} I(\boldsymbol{y}_A; q_A)$ associated with the kernel k, where $I(\boldsymbol{y}_A; q_A)$ denotes the mutual information between q evaluated at locations in the set A and the noisy samples $\boldsymbol{y}_A$ collected at A [14]. This definition lets us build upon the finite time convergence of COAT-MPC (Section VI).

B. Safe exploration

In this section, we introduce the necessary tools from prior works [6], [8] required to explore the domain of cost function weights efficiently while ensuring Eq. (2).

Optimistic, pessimistic and reachable sets. Utilizing the GP posterior Eq. (4), we construct intersecting lower and upper confidence bounds on q at each iteration $n \geq 1$ as:

$$\begin{aligned} l_n(\boldsymbol{\theta}) &\coloneqq \max\left(l_{n-1}(\boldsymbol{\theta}), \mu_{n-1}(\boldsymbol{\theta}) - \sqrt{\beta_n}\sigma_{n-1}(\boldsymbol{\theta})\right), \\ u_n(\boldsymbol{\theta}) &\coloneqq \min\left(u_{n-1}(\boldsymbol{\theta}), \mu_{n-1}(\boldsymbol{\theta}) + \sqrt{\beta_n}\sigma_{n-1}(\boldsymbol{\theta})\right), \end{aligned} \tag{5}$$

initialized with $l_0(\boldsymbol{\theta}) = \mu_0(\boldsymbol{\theta}) - \sqrt{\beta_1}\sigma_0(\boldsymbol{\theta})$ and $u_0(\boldsymbol{\theta}) = \mu_0(\boldsymbol{\theta}) + \sqrt{\beta_1}\sigma_0(\boldsymbol{\theta})$. Note that $l_n(\cdot)$ is non-decreasing and $u_n(\cdot)$ is non-increasing in n, i.e.,

$$l_{n+1}(\boldsymbol{\theta}) \geq l_n(\boldsymbol{\theta}), u_{n+1}(\boldsymbol{\theta}) \leq u_n(\boldsymbol{\theta}), \forall \boldsymbol{\theta} \in D,$$

directly by construction using intersecting confidence bounds in Eq. (5). Using this and the GPs error bounds from Theorem 2 of [27], we get the following corollary [25]:

Corollary 1 (Theorem 2 [27])**.** *Let Assumption 2 hold. If $\sqrt{\beta_n} = B + 4\sigma\sqrt{\gamma_n + 1 + \ln(1/\delta)}$, it holds that $l_n(\boldsymbol{\theta}) \leq q(\boldsymbol{\theta}) \leq u_n(\boldsymbol{\theta}), \forall \boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$ with probability at least $1 - \delta$.*

Throughout this work, we use $\sqrt{\beta_n}$ from Corollary 1. Similar to GOOSE [6], we define a one-step reachability operator exploiting the L-Lipschitz continuity of q and build pessimistic and optimistic constraint operators over it using the derived lower and upper confidence bounds.

$$\begin{aligned} r^\epsilon(\mathcal{S}) &= \{\boldsymbol{\theta} \in D \,|\, \exists \boldsymbol{\theta}' \in \mathcal{S} : q(\boldsymbol{\theta}') - Ld(\boldsymbol{\theta}, \boldsymbol{\theta}') - \epsilon \geq \tau\} \\ p_n(\mathcal{S}) &= \{\boldsymbol{\theta} \in D \,|\, \exists \boldsymbol{\theta}' \in \mathcal{S} : l_n(\boldsymbol{\theta}') - Ld(\boldsymbol{\theta}, \boldsymbol{\theta}') \geq \tau\} \\ o_n^\epsilon(\mathcal{S}) &= \{\boldsymbol{\theta} \in D \,|\, \exists \boldsymbol{\theta}' \in \mathcal{S} : u_n(\boldsymbol{\theta}') - Ld(\boldsymbol{\theta}, \boldsymbol{\theta}') - \epsilon \geq \tau\}. \end{aligned}$$

A visual representation of the pessimistic and optimistic operators evaluated at a single point is depicted in Fig. 2. For notational convenience, we denote $r(\mathcal{S}) \coloneqq r^0(\mathcal{S})$ when referring to $\epsilon = 0$ case (analogously for the pessimistic and the optimistic operator as well). By applying these one-step constraint operators recursively, we next define the pessimistic, optimistic and reachability expansion operators, which are used to obtain the pessimistic and optimistic

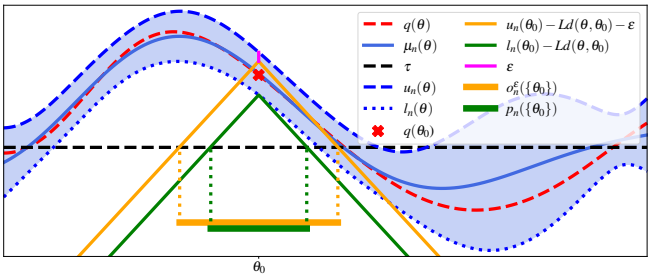


Fig. 2: Pessimistic and optimistic operators evaluated at θ_0. The operators make use of the GP upper and lower confidence bounds, as well as the L-Lipschitz continuity. In this example, $d(\theta, \theta_0)$ is the Euclidean distance function, where θ_0 is fixed. Additionally, τ is set arbitrarily.

estimates of the true constraint set:

$$\tilde{R}^\epsilon(\mathcal{S}) = \lim_{m\to\infty} R^{\epsilon,m}(\mathcal{S}), \tag{6}$$

$$\tilde{P}_n(\mathcal{S}) = \lim_{m\to\infty} P_n^m(\mathcal{S}), \tag{7}$$

$$\tilde{O}_n^\epsilon(\mathcal{S}) = \lim_{m\to\infty} O_n^{\epsilon,m}(\mathcal{S}), \tag{8}$$

where $P_n^m(\mathcal{S}) := p_n(p_n \cdots (p_n(\mathcal{S})))$ and $O_n^{\epsilon,m}(\mathcal{S}) := o_n^\epsilon(o_n^\epsilon \cdots (o_n^\epsilon(\mathcal{S})))$ are the m-step pessimistic and optimistic expansion operators. Using the expansion operators on $\mathcal{S}_{n-1}^p$ we obtain pessimistic $\mathcal{S}_n^p = \tilde{P}_n(\mathcal{S}_{n-1}^p)$ and optimistic $\mathcal{S}_n^{o,\epsilon} = \tilde{O}_n^\epsilon(\mathcal{S}_{n-1}^p)$ estimates of the true constraint set. Analogously, $R^{\epsilon,m}(\mathcal{S}) := r^\epsilon(r^\epsilon \cdots (r^\epsilon(\mathcal{S})))$ denotes the m-step $\epsilon-$close true reachability operator. Applying the reachability operator from Eq. (6) on the initial seed $\mathcal{S}_0$, we obtain the $\epsilon-$close true reachability set $\mathcal{S}^{q,\epsilon} = \tilde{R}^\epsilon(\mathcal{S}_0)$, which includes all the parameters being at least $\epsilon-$conservative from violating the constraint.

V. COAT-MPC

In this section, we present COAT-MPC, for the optimization of MPC cost function parameters while respecting the performance constraint. The algorithm is introduced in Algorithm 2 with its optimality guarantees deferred to Section VI.

Intuition. Our algorithm's goal is (i) to ensure the satisfaction of Eq. (2) while (ii) converging to the optimal cost function weights with few tuning iterations. For the former part (i), we sample weights from the pessimistic set $\mathcal{S}_n^p$, which guarantees satisfying Eq. (2) with high probability. To ensure the later part (ii), we set a goal $\boldsymbol{\theta}_n^g$ in the optimistic set $\mathcal{S}_n^{o,\epsilon}$, outside of which the weights do not satisfy Eq. (2). Additionally, to converge to the optimal weights with fewer tuning iterations, we employ a goal-directed approach and use an expansion method (see Algorithm 1) towards this goal. We showcase a visual representation of COAT-MPC in the 1D setting in Fig. 3.

Algorithm 1 Constrained Expansion (CE)

1: **Input:** $\mathcal{S}_n^p, \boldsymbol{\theta}_n^g$
2: **Recommend:** $\arg\min_{\boldsymbol{\theta}\in\mathcal{S}_n^p} ||\boldsymbol{\theta}_n^g - \boldsymbol{\theta}||_2$, s.t. $w_n(\boldsymbol{\theta}) \geq \epsilon$

Constrained Expansion. The objective is to learn about the satisfaction of Eq. (2) of the current goal $\boldsymbol{\theta}_n^g$, given the pessimistic set $\mathcal{S}_n^p$. COAT-MPC's expansion strategy recommends the closest point to the goal, with respect to the Euclidean distance, inside the pessimistic set and that is not ϵ-accurate, i.e., the width of the confidence bounds $w_n(\boldsymbol{\theta}) = u_n(\boldsymbol{\theta}) - l_n(\boldsymbol{\theta})$ evaluated at the point is greater than or equal to ϵ. Thus, COAT-MPC recommends parameters that satisfy the constraint and that are as close as possible to the goal while maintaining exploration through the statistical confidence ϵ. If the algorithm is certain enough about the performance q of a parameter, it will not further explore it.

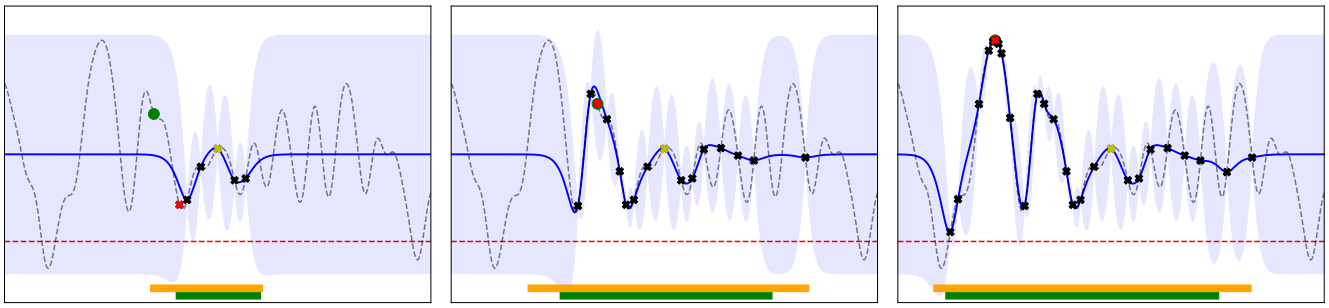
Algorithm 2 COAT-MPC

1: **Input:** Initial seed $\mathcal{S}_0$, $q \sim \mathcal{GP}(\mu_0(\boldsymbol{\theta}), k_0(\boldsymbol{\theta}, \boldsymbol{\theta}'))$, τ, D, Lipschitz constant L
2: $\mathcal{S}_0^p \leftarrow \mathcal{S}_0$, $\mathcal{S}_0^{o,\epsilon} \leftarrow D$
3: **for** $n = 1, \ldots N_{max}$, **do**
4: $\quad\boldsymbol{\theta}_n^g \leftarrow \arg\max_{\boldsymbol{\theta}\in\mathcal{S}_{n-1}^{o,\epsilon}} u_{n-1}(\boldsymbol{\theta})$
5: $\quad$**if** $\boldsymbol{\theta}_n^g \in \mathcal{S}_{n-1}^p$ and $w_{n-1}(\boldsymbol{\theta}_n^g) < \epsilon$ **then**
6: $\quad\quad$Terminate
7: $\quad$**else if** $\boldsymbol{\theta}_n^g \notin \mathcal{S}_{n-1}^p$ **then**
8: $\quad\quad\boldsymbol{\theta}_n \leftarrow \text{CE}(\mathcal{S}_{n-1}^p, \boldsymbol{\theta}_n^g)$
9: $\quad$**end if**
10: $\quad y_n \leftarrow q(\boldsymbol{\theta}_n) + \eta_n$ and Update GP
11: $\quad\mathcal{S}_n^p \leftarrow \tilde{P}_n(\mathcal{S}_{n-1}^p)$
12: $\quad\mathcal{S}_n^{o,\epsilon} \leftarrow \tilde{O}_n^\epsilon(\mathcal{S}_{n-1}^p)$
13: **end for**
14: **Recommend:** $\boldsymbol{\theta}_n^g$

COAT-MPC. The pseudocode is summarized in Algorithm 2. We start the algorithm from parameters within $\mathcal{S}_0$, where the constraint $q(\boldsymbol{\theta}) \geq \tau, \forall \boldsymbol{\theta} \in \mathcal{S}_0$ is known to be satisfied due to Assumption 1. In Line 2, we initialize the pessimistic set to the initial seed $\mathcal{S}_0$ and the optimistic set to the parameters' domain D. We next compute the goal $\boldsymbol{\theta}_n^g$ within the optimistic set using the Upper Confidence Bounds (UCB) (Line 4). The objective is to reach the goal while ensuring that sufficient information is gained for exploration. This is measured using the width of the confidence bounds $w_{n-1}(\boldsymbol{\theta})$. Based on the location of the goal and the uncertainty about it, COAT-MPC considers the following two cases:

1) **The goal is in the pessimistic set with the uncertainty below ϵ**, i.e., $w_{n-1}(\boldsymbol{\theta}_n^g) < \epsilon$: This case represents that the goal satisfies Eq. (2) and the performance function value is known up to the desired confidence. Since the goal defined using UCB criteria is an upper bound over the possible locations that fulfill the constraint, we terminate the algorithm (Line 6) with desired guarantees (Section VI).
2) **The goal is not in the pessimistic set**: In order to expand the pessimistic set to the goal location we use *constrained expansion* defined in Algorithm 1 (Line 8) and obtain a sampling location. Then we collect a noisy measurement of q and update the posterior (Line 10)..

If none of the two cases holds, the goal is within the pessimistic set but the uncertainty $w_{n-1}(\boldsymbol{\theta}_n^g) \geq \epsilon$. In this case, we directly collect a noisy measurement of q to reduce the uncertainty, update the posterior and continue the process



(a) COAT-MPC at $n = 5$ (b) COAT-MPC at $n = 15$ (c) COAT-MPC at $n = 28$. Final iteration.

Fig. 3: COAT-MPC illustration. (i) The grey, dashed line represents the true function. (ii) The red, dashed line represents the constraint. (iii) The blue line represents the Gaussian Process mean, and the shaded blue area represents the confidence bounds ($\mu_n(\boldsymbol{\theta}) \pm \sqrt{\beta_n}\sigma_n(\boldsymbol{\theta})$). (iv) The cross markers represent the samples, with yellow denoting the first sample and red the COAT-MPC recommended sample. (v) The green dot denotes the goal at each iteration. The algorithm learns the pessimistic (green bar) and optimistic (orange bar) sets, and explores the parameter space while satisfying the performance constraint. At $n = 5$, the goal is outside of the pessimistic set, although it is inside the optimistic set. COAT-MPC expands the pessimistic set by approaching the goal. It reaches the goal at $n = 15$, and by further expanding the sets, it discovers the maximum of the function at $n = 28$.

to get the new goal location (Line 10). Finally, if the algorithm has not terminated, the pessimistic and optimistic sets are updated using their expansion operators (Lines 11 and 12). This process expands the potential set of parameters that fulfill Eq. (2) and explores the parameter domain.

VI. Theoretical Analysis

In this section, we present our core theoretical result, i.e., convergence to optimal tuning parameters while satisfying Eq. (2) in finite time with arbitrarily high probability. We first make the following assumption for the finite sample complexity result.

Assumption 3. *$\beta_n\gamma_n$ grows sublinear in n, i.e., $\beta_n\gamma_n < \mathcal{O}(n)$.*

This assumption is common in most prior works [9], [7], [6] aimed to establish sample complexity or sublinear regret results and are not restrictive. It can be satisfied for commonly used kernels, e.g., linear kernels, squared exponential, Matérn, etc., with sufficient eigen decay [28], [14] under the bounded B_q Assumption 2.

Theorem 1. *Let Assumptions 1 to 3 hold and $n^\star$ be the largest integer such that $\frac{n^\star}{\beta_{n^\star}\gamma_{n^\star}} \leq \frac{C_1}{\epsilon^2}$ with $C_1 := 8/\log(1+\sigma_\eta^{-2})$. With probability at least $1-\delta$,* COAT-MPC *satisfies* (2) *for the evaluated parameters $\boldsymbol{\theta}_n, \forall n \geq 1$ and the resulting closed-loop system* (1) *satisfies state and input constraints for all times $t \geq 0$. Moreover, $\exists n \leq n^\star$ satisfying,*

$$q(\boldsymbol{\theta}_n) \geq \max_{\boldsymbol{\theta} \in \mathcal{S}^{q,\epsilon}} q(\boldsymbol{\theta}) - \epsilon.$$

with probability at least $1-\delta$.

The proof is in Appendix A. Thus, COAT-MPC guarantees satisfying Eq. (2) with high probability at each tuning iteration. Moreover, it ensures finite time convergence to the reachable optimal tuning parameters under performance constraints. Intuitively, Eq. (2) is ensured by executing the parameters from the pessimistic set; while the other state and input constraints are ensured directly by MPC (see Eq. (1)). Since the final recommended tuning parameter is as per UCB (optimistic estimate of performance) with uncertainty below ϵ, we are guaranteed to converge to the optimal solution reachable under performance constraints. In contrast to the earlier sample complexity results [8] in discrete domains, we do not have an explicit dependence on the domain size $|\tilde{R}^\epsilon(\mathcal{S})|$. This makes our bound tighter and more usable for larger domains (or finer discretization). We achieved this result by extending the sample complexity analysis of [7] from continuous domains to discrete domains.

VII. Experimental Results

We present an extensive evaluation of COAT-MPC in an autonomous racing application. Our evaluation is conducted using an autonomous racing simulation and the scaled RC racecar platform shown in Fig. 4. We first discuss the particular MPC formulation used in our experiments in Section VII-A, followed by our experimental setup in Section VII-B. Finally, we present the simulation and experimental results in Sections VII-C and VII-D, respectively.

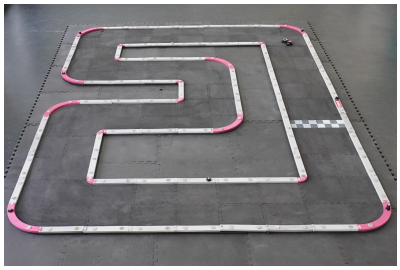


Fig. 4: 1:28 scale RC racecar [11] and track used in the experiments.

A. Model predictive contouring control (MPCC)

In the proposed experiments with the autonomous racing platform, we use a particular MPC formulation commonly denoted as MPCC (Model Predictive Counturing Control) [10], which has been successfully applied for autonomous racing in a known track. In particular, our MPCC is formulated as:

$$\begin{aligned} \min_{\boldsymbol{u}_{0:N}} \quad & \sum_{i=0}^{N} -q_\lambda \lambda_i + \boldsymbol{e}_i^T \boldsymbol{Q} \boldsymbol{e}_i + \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i \\ \text{s.t.} \quad & \boldsymbol{x}_0 = \hat{\boldsymbol{x}}(t),\ \boldsymbol{x}_{i+1} = \boldsymbol{f}_d(\boldsymbol{x}_i, \boldsymbol{u}_i) \\ & \boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{u}_i \in \mathcal{U},\ \boldsymbol{g}_i(\boldsymbol{x}_i, \boldsymbol{u}_i) \in \mathcal{G}, \end{aligned} \tag{9}$$

Algorithm	Simulation				RC platform			
	#Constraint violations	Min. lap time[s]	Mean lap time ± std. deviation[s]	Number of iterations	#Constraint violations	Min. lap time[s]	Mean lap time ± std. deviation[s]	Number of iterations
COAT-MPC	**0**	**4.68**	5.57 ± 0.57	**28**	**0.33**	**6.49**	7.55 ± 0.50	**21.66**
SAFEOPT	1.8	4.71	5.84 ± 0.82	70	5.33	6.52	7.88 ± 0.74	70
GP-UCB	5.2	**4.68**	$\mathbf{5.03 \pm 1.02}$	70	7.0	6.82	$\mathbf{7.51 \pm 0.77}$	70
WML	2.83	4.71	5.30 ± 0.80	70	6.66	6.95	7.60 ± 0.68	70
EI_C	7.0	**4.68**	5.24 ± 1.16	70	-	-	-	-
CRBO	16.4	4.71	5.55 ± 1.39	70	-	-	-	-

TABLE I: Averaged results of COAT-MPC and baselines. The algorithms were evaluated 5 times in simulation and 3 times with the RC platform.

where $\boldsymbol{x}_i = [x_i, y_i, \psi_i, v_{x_i}, v_{y_i}, \dot{\psi}_i]$ is the state of the car (including position, orientation, and velocities), $\boldsymbol{u}_i = [\delta_i, T_i]$ is the control input (steering angle and drivetrain command), λ is the parameter that determines the progress along the reference trajectory, $\boldsymbol{e}_i$ denotes contour and lag errors, $\boldsymbol{f}_d(\cdot,\cdot)$ is the nominal car model consisting of a Pacejka dynamic bicycle model [29], and $\boldsymbol{g}_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$ are linear and nonlinear constraints on the states and inputs. The matrix $\boldsymbol{Q} = \text{diag}([Q_{contour}, Q_{lag}])$ controls the longitudinal and lateral deviation from the reference trajectory, q_λ regulates the progress of the car, and $\boldsymbol{R}$ determines the smoothness of the inputs.

This MPC aims to maximize the progress while penalizing the deviation from the reference trajectory. It is worth noting that this MPC formulation does not directly minimize time. However, we set lap time as the performance function of COAT-MPC to achieve lap time minimization.

B. Experimental setup

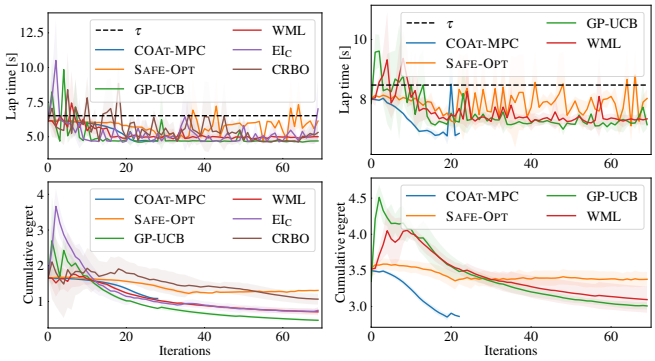
To quantify the performance of the MPC, we define the performance function q as the negative lap time of a single flying lap, where the car does not start from a stationary position. The negative sign is introduced to reflect the objective of minimizing lap time. Additionally, we establish the performance upper bound as $\tau = \tau_{scale}(q(\mathcal{S}_0[0]) + \eta)$, where $\tau_{scale} \geq 1$ is a user-specified scaling factor, and $q(\mathcal{S}_0[0]) + \eta$ represents a noisy evaluation of the negative lap time, due to different errors while running in the real-world, e.g., state estimation or process noises. This noisy laptime is obtained using the initial seed parameters, which are known a priori from manual tuning. Hence, we constrain the lap time to always be lower than the initial lap time multiplied by a scaling factor that is larger than one.

We conduct a comparative analysis of our proposed method with several unconstrained methods, namely, GP-UCB [14], Weighted Maximum Likelihood (WML) [13], and Confidence Region Bayesian Optimization (CRBO) [18]. Additionally, we evaluate our method against constrained optimization methods, specifically, (EI_C) [22] and SAFEOPT [8].

In our experiments, we jointly optimize $Q_{contour}$ and Q_{lag} (see Eq. (9)). We uniformly discretize the parameters' domain into 10,000 combinations within the range of $[0, 1000]^2$. These combinations are normalized to $[0, 1]^2$. Furthermore, we set the initial weights of $Q_{contour}$ and Q_{lag} to 500. For the methods that use a Gaussian Process to model the performance function, we select a Matérn Kernel with a smoothness parameter of $\nu = 5/2$. A unique length-scale of $l = 0.1$ is chosen for both dimensions and we use $\beta = 5.0$.

C. Simulation results

We present a comprehensive evaluation of our method in comparison to the baselines over a total of 70 iterations, during which the methods are permitted to sample and assess 70 distinct parameters. Note that, in this setup, COAT-MPC takes less than 70 iterations due to its termination criteria. As presented in Table I and Fig. 5, our method outperforms baselines in terms of performance constraint violations while converging to the optimal parameters in 30 iterations. We observe that COAT-MPC achieves the same lap time as GP-UCB, indicating that both algorithms converge to the optimal parameters. However, GP-UCB shows slightly better cumulative regret over the initial 30 iterations (see Fig. 5a). This is mainly due to the simulation domain D, where a large part satisfies Eq. (2) (see Fig. 6) and results in lower lap times. As a result, GP-UCB samples parameters without significantly violating constraints, resulting in a better lap time and cumulative regret. Although cumulative regret indicates convergence, our focus is on achieving the best lap time as quickly as possible without violating Eq. (2) and in all these respects, COAT-MPC outperforms the baselines.



(a) Simulation results. (b) RC platform results.

Fig. 5: Lap time and cumulative regret over time with their standard deviation. COAT-MPC converges in only 20-30 iterations and achieves the lowest cumulative regret over time in the RC platform experiments.

Fig. 6 illustrates the tuning process of our method and baselines. COAT-MPC starts by sampling close to the initial seed, aiming to expand the pessimistic set. Subsequently, our method cautiously samples parameters that result in an improved lap time, and converges once it is ϵ-certain that it has found the optimal parameters. As shown, COAT-MPC performs a goal-directed exploration to converge to the optimal parameters. Our method requires less exploration to converge compared to the baselines and always satisfies the performance constraint.

D. RC platform results

After observing the outcomes in simulation, our algorithm is benchmarked against SAFEOPT, GP-UCB, and WML, which proved to be the best among all baselines in terms of constraint violations. As shown in Table I, our approach surpasses the baselines in terms of constraint violations and closely approximates the average mean lap time of GP-UCB, while effectively converging to the optimal parameters. Note that COAT-MPC violates Eq. (2) once in our experiments. During our experiments with the RC platform, various external factors such as motion capture and state estimation noise influenced performance, potentially causing the car to crash and violate the constraints. The COAT-MPC's constraint violation could be attributed to one of these external factors.

Our method stands out by achieving the lowest cumulative regret over time and converging to the optimal parameters in just 20-25 iterations, as shown in Fig. 5 and Table I. Additionally, as demonstrated in the regret plot of Fig. 5, COAT-MPC achieves a lower minimum regret as compared to the baselines, which indicates that our method is capable of converging to the optimal set of parameters while satisfying Eq. (2) with high probability. Fig. 6 illustrates the tuning process of our method compared to the baselines.

VIII. CONCLUSIONS

We propose COAT-MPC, a method for MPC tuning that guarantees a performance constraint satisfaction with arbitrarily high probability. Our approach leverages the assumption of Lipschitz continuity in the objective function to construct pessimistic and optimistic constraint sets. We use the optimistic set to define a goal location at each iteration, while we restrict our recommendations to be within the pessimistic set. We present a theoretical analysis of our method, conclusively demonstrating its ability to achieve optimal tuning parameters in finite time while guaranteeing the satisfaction of the performance constraint with arbitrarily high probability. Additionally, our evaluation against state-of-the-art methods demonstrates that our method outperforms them in terms of the number of constraint violations, as well as in cumulative regret over time, in the context of MPC tuning for autonomous racing. Finally, an interesting line of future work could be to extend the method to work in high-dimensional parameter space. Our approach discretizes the parameter space in order to compute the pessimistic and optimistic sets. This discretization process, however, triggers exponential growth in memory consumption, making the method unfeasible for a large number of parameters.

APPENDIX

A. Auxiliary lemmas for Proof of Theorem 1

In this section, we develop tools for the theoretical analysis of COAT-MPC, present auxiliary lemmas, and finally use them to prove Theorem 1.

To simplify analysis, we define a pessimistic set, $\mathcal{S}_n^{p,sage} := \{\boldsymbol{\theta} \in D | \exists \boldsymbol{\theta}' \in D, l_n(\boldsymbol{\theta}') - L_q d(\boldsymbol{\theta}, \boldsymbol{\theta}') \geq \tau\}$, motivated from [7]. Note that $\mathcal{S}_n^{p,sage}$ does not explicitly depend on the initial safe set and is defined over the domain, precisely, $\mathcal{S}_n^{p,sage} = p_n(D)$. The following lemma establishes that our pessimistic set, $\mathcal{S}_n^p$ is subset of the one in [7].

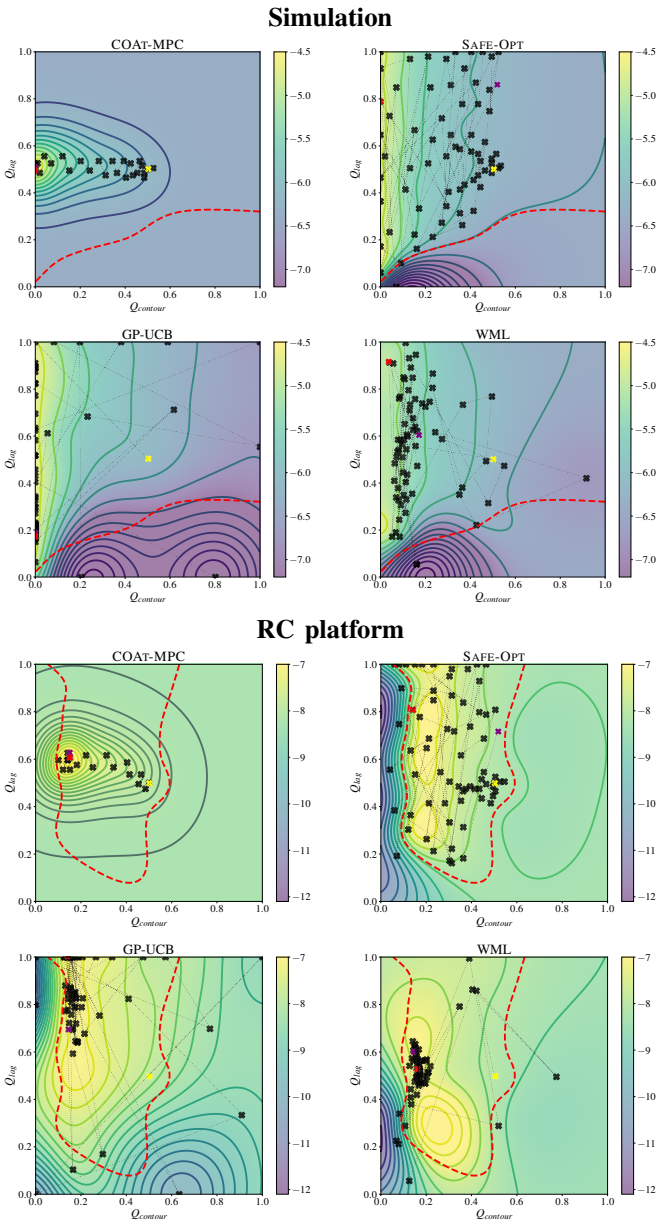


Fig. 6: Method's tuning process. The black marker denotes the method's samples. The black lines denote the sample trajectories. The yellow marker denotes the initial values of $Q_{contour}$ and Q_{lag}, the purple marker is the last sample, and the red marker is the best lap time's sample. The red line denotes the area where $q(\boldsymbol{\theta}) = \tau$. The heatmap is the GP posterior mean.

Lemma 1. $\mathcal{S}_n^p \subseteq \mathcal{S}_n^{p,sage}, \forall n \geq 0$.

Proof. Note that $\mathcal{S}_n^p = \tilde{P}_n(\mathcal{S}_{n-1}^p)$ and using (7), we get, $\mathcal{S}_n^p = \lim_{m\to\infty} P_n^m(\mathcal{S}_{n-1}^p)$. This implies $\mathcal{S}_n^p = p_n(\lim_{m\to\infty} P_n^m(\mathcal{S}_{n-1}^p)) \subseteq p_n(D) = \mathcal{S}_n^{p,sage}$. The last equality follows by definition of the set, $\mathcal{S}_n^{p,sage}$. □

Corollary 2 (Theorem 1 [7])**.** *Let Assumptions 2 and 3 hold and $n^\star$ be the largest integer satisfying $\frac{n^\star}{\beta_{n^\star}\gamma_{n^\star}} \leq \frac{C}{\epsilon^2}$, with $C = 8/\log(1+\sigma_q^{-2})$. The sampling scheme $\boldsymbol{\theta}_n \in \mathcal{S}_{n-1}^p$: $w_{n-1}(\boldsymbol{\theta}_n) \geq \epsilon$ satisfy $q(\boldsymbol{\theta}_n) \geq \tau, \forall n \geq 1$ with probability at least $1-\delta$ and $\exists n \leq n^\star : \forall \boldsymbol{\theta} \in \mathcal{S}_n^p,\ w_n(\boldsymbol{\theta}) < \epsilon$.*

Proof. In [7], SageMPC uses a sampling rule $\boldsymbol{\theta}_n \in \mathcal{S}_{n-1}^{p,sage}$: $w_{n-1}(\boldsymbol{\theta}_n) \geq \epsilon$ which aligns with our sampling rule of $w_{n-1}(\boldsymbol{\theta}_n) \geq \epsilon$ (Line 10). Moreover, Theorem 1 [7], i.e, $\exists n \leq n^\star : \forall \boldsymbol{\theta} \in \mathcal{S}_n^{p,sage},\ w_n(\boldsymbol{\theta}) < \epsilon$, and Lemma 1, i.e., $\mathcal{S}_{n-1}^{p} \subseteq \mathcal{S}_{n-1}^{p,sage}$ implies $\exists n \leq n^\star : \forall \boldsymbol{\theta} \in \mathcal{S}_n^{p},\ w_n(\boldsymbol{\theta}) < \epsilon$, which ensures finite time convergence.

Next we prove satisfaction of performance constraint (2). Note that, $\forall \boldsymbol{\theta} \in \mathcal{S}_n^p \subseteq \mathcal{S}_n^{p,sage} \implies \exists \boldsymbol{\theta}' \in D : l_n(\boldsymbol{\theta}') - L_q d(\boldsymbol{\theta}, \boldsymbol{\theta}') \geq \tau \implies q(\boldsymbol{\theta}) \geq \tau$ with probability at least $1-\delta$ using Corollary 1. □

Thus, using Theorem 1 [7], COAT-MPC ensures high probability satisfaction of (2) at every sampling location and guarantees termination of the process within $n^\star$ iterations. Next, we prove the optimality guarantees for COAT-MPC.

Lemma 2. *Let Assumption 2 holds and* $w_{n-1}(\boldsymbol{\theta}_n^g) < \epsilon$, *where,* $\boldsymbol{\theta}_n^g \coloneqq \arg\max_{\boldsymbol{\theta} \in \mathcal{S}_{n-1}^{o,\epsilon}} u_{n-1}(\boldsymbol{\theta})$. *Then with probability at least* $1-\delta$ *it holds that,* $q(\boldsymbol{\theta}_n^g) \geq \max_{\boldsymbol{\theta} \in \mathcal{S}^{q,\epsilon}} q(\boldsymbol{\theta}) - \epsilon$.

Proof. By construction of confidence bounds (5), it follows that $q(\boldsymbol{\theta}) \leq u_{n-1}(\boldsymbol{\theta}), \forall n \geq 1, \boldsymbol{\theta} \in D$ with probability at least $1-\delta$. This implies $\mathcal{S}^{q,\epsilon} \coloneqq \tilde{R}^\epsilon(\mathcal{S}_0) \subseteq \mathcal{S}_{n-1}^{o,\epsilon}$. Next, given $w_{n-1}(\boldsymbol{\theta}_n^g) < \epsilon$ implies $l_{n-1}(\boldsymbol{\theta}_n^g) > u_{n-1}(\boldsymbol{\theta}_n^g) - \epsilon \geq q(\boldsymbol{\theta}_n^g) - \epsilon$.

Define $\hat{\boldsymbol{\theta}} \coloneqq \arg\max_{\boldsymbol{\theta} \in \mathcal{S}^{q,\epsilon}} q(\boldsymbol{\theta})$ and using both the above derived inequalities, we get,

$$\begin{aligned} u_{n-1}(\hat{\boldsymbol{\theta}}) &\leq u_{n-1}(\boldsymbol{\theta}_n^g) \qquad (\mathcal{S}^{q,\epsilon} \coloneqq \tilde{R}^\epsilon(\mathcal{S}_0) \subseteq \mathcal{S}_{n-1}^{o,\epsilon}) \\ &< l_{n-1}(\boldsymbol{\theta}_n^g) + \epsilon \leq q(\boldsymbol{\theta}_n^g) + \epsilon \quad (w_{n-1}(\boldsymbol{\theta}_n^g) < \epsilon) \\ \implies q(\hat{\boldsymbol{\theta}}) &\leq q(\boldsymbol{\theta}_n^g) + \epsilon \end{aligned}$$

This implies, $q(\boldsymbol{\theta}_n^g) \geq \max_{\boldsymbol{\theta} \in \mathcal{S}^{q,\epsilon}} q(\boldsymbol{\theta}) - \epsilon$. □

Next, we prove our main theorem using the lemmas above, and additionally guarantee the satisfaction of state and input constraints of the closed-loop system throughout the process.

Proof of Theorem 1. The initial seed (Assumption 1) ensures (2), which implies that MPC is feasible and thus satisfies state and input constraints at $n = 0$. In COAT-MPC, we sample at $\boldsymbol{\theta}_n \in \mathcal{S}_{n-1}^p \implies q(\boldsymbol{\theta}_n) \geq \tau$ (Corollary 2) under Assumption 2, which ensures feasibility of the resulting closed-loop system (1) $\forall n \geq 1$.

Finite time convergence guarantees follows from Corollary 2 which provides a sample complexity bound, i.e., $\exists n \leq n^\star$ under Assumptions 2 and 3 before which COAT-MPC will terminate.

Since COAT-MPC samples only if $w_{n-1}(\boldsymbol{\theta}_n^g) > \epsilon$ and Corollary 2 implies $\exists n \leq n^\star$ under which uncertainty in the $\mathcal{S}_{n-1}^p$ is uniformly bounded by ϵ. This implies $w_{n-1}(\boldsymbol{\theta}_n^g) \leq \epsilon$ and using this, Lemma 2 establish the resulting optimality of the COAT-MPC algorithm. □

Note that our proving strategy differs from that of GOOSE and the SageMPC work. We employ a sampling strategy motivated by SageMPC; however, because we operate in a discrete domain, we define our sets using GOOSE's definitions. Specifically, we do this to relate the growth of the pessimistic set with respect to the initial seed of the agent as in GOOSE while being able to use more efficient sample complexity bounds from [7].

REFERENCES

[1] J. Kabzan, M. I. Valls, V. J. Reijgwart, H. F. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, *et al.*, "Amz driverless: The full autonomous racing system," *Journal of Field Robotics*, 2020.

[2] D. Kang, F. De Vincenti, N. C. Adami, and S. Coros, "Animal motions on legged robots using nonlinear model predictive control," *IROS*, 2022.

[3] S. Kuindersma, R. Deits, M. F. Fallon, A. K. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, 2015.

[4] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning," *MIT Press*, 2006.

[5] F. Berkenkamp, A. Krause, and A. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *Machine Learning*, 2021.

[6] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration for interactive machine learning," *NeuRIPS*, 2019.

[7] M. Prajapat, J. Köhler, M. Turchetta, A. Krause, and M. N. Zeilinger, "Safe guaranteed exploration for non-linear systems," *ArXiv preprint*, 2024.

[8] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," *ICML*, 2015.

[9] M. Prajapat, M. Turchetta, M. Zeilinger, and A. Krause, "Near-optimal multi-agent learning for safe coverage control," *NeuRIPS*, 2022.

[10] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control App. and Methods*, 2015.

[11] A. Carron, S. Bodmer, L. Vogel, R. Zurbrügg, D. Helm, R. Rickenbach, S. Muntwiler, J. Sieber, and M. N. Zeilinger, "Chronos and crs: Design of a miniature car-like robot and a software framework for single and multi-agent robotics and control," *ICRA*, 2023.

[12] A. Loquercio, A. Saviolo, and D. Scaramuzza, "Autotune: Controller tuning for high-speed flight," *RA-L*, 2022.

[13] A. Romero, S. Govil, G. Yilmaz, Y. Song, and D. Scaramuzza, "Weighted maximum likelihood for controller tuning," *ICRA*, 2022.

[14] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *ICML*, 2009.

[15] P. Frazier, "A tutorial on bayesian optimization," *ArXiv*, 2018.

[16] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*, ser. Mathematics and its Applications. Springer Netherlands, 2012.

[17] L. P. Fröhlich, C. Küttel, E. Arcari, L. Hewing, M. N. Zeilinger, and A. Carron, "Contextual tuning of model predictive control for autonomous racing," *IROS*, 2022.

[18] L. P. Fröhlich, M. N. Zeilinger, and E. D. Klenske, "Cautious bayesian optimization for efficient and scalable policy search," *L4DC*, 2021.

[19] "Scalable global optimization via local Bayesian optimization," *Advances in Neural Information Processing Systems*, NeuRIPS.

[20] M. Prajapat, M. Mutnỳ, M. N. Zeilinger, and A. Krause, "Submodular reinforcement learning," *arXiv preprint arXiv:2307.13372*, 2023.

[21] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J.-B. Mouret, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *T-RO*, 2018.

[22] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints," *ICML*, 2014.

[23] J. T. Wilson, F. Hutter, and M. P. Deisenroth, "Maximizing acquisition functions for bayesian optimization," *NeuRIPS*, 2018.

[24] F. Sorourifar, G. Makrygirgos, A. Mesbah, and J. A. Paulson, "A data-driven automatic tuning method for mpc under uncertainty using constrained bayesian optimization," *ADCHEM*, 2021.

[25] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," *ICRA*, 2016.

[26] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2018.

[27] S. R. Chowdhury and A. Gopalan, "On kernelized multi-armed bandits," *ICML*, 2017.

[28] S. Vakili, K. Khezeli, and V. Picheny, "On information gain and regret bounds in gaussian process bandits," *AISTATS*, 2021.

[29] H. B. Pacejka, in *Tyre and Vehicle Dynamics (Second Edition)*. Oxford: Butterworth-Heinemann, 2006.