Artificial Intelligence 320 (2023) 103922

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

# GoSafeOpt: Scalable safe exploration for global optimization of dynamical systems[☆]

Bhavya Sukhija[a,*], Matteo Turchetta[a], David Lindner[a], Andreas Krause[a], Sebastian Trimpe[b], Dominik Baumann[c,d]

[a] Department of Computer Science, ETH Zürich, Switzerland
[b] Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Germany
[c] Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland
[d] Department of Information Technology, Uppsala University, Sweden

A R T I C L E   I N F O

*Article history:*
Received 31 March 2022
Received in revised form 12 April 2023
Accepted 14 April 2023
Available online 20 April 2023

*Keywords:*
Model-free learning
Bayesian optimization
Safe learning

A B S T R A C T

Learning optimal control policies directly on physical systems is challenging. Even a single failure can lead to costly hardware damage. Most existing model-free learning methods that guarantee safety, i.e., no failures, during exploration are limited to local optima. This work proposes GoSafeOpt as the first provably safe and optimal algorithm that can safely discover globally optimal policies for systems with high-dimensional state space. We demonstrate the superiority of GoSafeOpt over competing model-free safe learning methods in simulation and hardware experiments on a robot arm.

© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

The increasing complexity of modern dynamical systems often makes deriving mathematical models for traditional model-based control approaches forbiddingly involved and time-consuming. Model-free reinforcement learning (RL) methods [1] are a promising alternative as they learn control policies directly from data. To succeed, they need to explore the system and its environment. Without a model, this can be risky and unsafe. Since modern hardware such as robots are expensive and their repairs are time-consuming, safe exploration is crucial to apply model-free RL in real-world problems. This paper proposes GoSafeOpt, a model-free learning algorithm that can search for globally optimal policies while guaranteeing safe exploration with high probability.

### 1.1. Related work

Advances in machine learning have motivated the usage of model-free RL algorithms for obtaining control policies [2–6]. However, directly applying these methods to policy optimization presents two major challenges: *(i)* Machine learning algorithms often require large amounts of data. In learning control, such data is often gathered by conducting experiments

☆ This paper is part of the Special Issue: “Risk-aware Autonomous Systems: Theory and Practice”.
* Corresponding author.
*E-mail addresses:* bhavya.sukhija@inf.ethz.ch (B. Sukhija), matteo.turchetta@inf.ethz.ch (M. Turchetta), david.lindner@inf.ethz.ch (D. Lindner), krausea@ethz.ch (A. Krause), trimpe@dsme.rwth-aachen.de (S. Trimpe), dominik.baumann@aalto.fi (D. Baumann).

https://doi.org/10.1016/j.artint.2023.103922
0004-3702/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

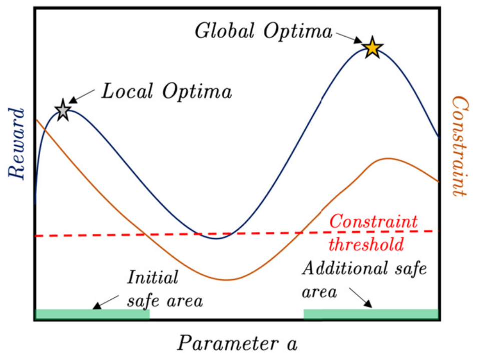

**Fig. 1.** Illustrative example with disjoint safe regions in the policy space. *The blue line depicts the objective, and the orange line is the constraint function. There are two safe regions that are marked in green.* SAFEOPT *cannot explore the global optimum if it is initialized in the left region.* (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

with physical systems, which is time-consuming and wears out the hardware. *(ii)* Learning requires exploration, which can lead to unwarranted and unsafe behaviors.

Challenges *(i)* and *(ii)* can be addressed jointly by Bayesian optimization (BO) with constraints. BO [7] is a class of black-box global optimization algorithms, that has been used in a variety of works [8–11] to optimize controllers in a sample-efficient manner. In constrained BO, there are two main classes of methods. On the one hand, approaches like [12–15] find safe solutions but allow unsafe evaluations during training. Herein, we focus on approaches that guarantee safety at all times during exploration, which is crucial when dealing with expensive hardware. SAFEOPT [16] and safe learning methods that emerged from it, e.g., [17–19], guarantee safe exploration with high probability by exploiting properties of the constraint functions, e.g., regularity. Unfortunately, these methods are limited to exploring a safe set connected with a known initial safe policy. Therefore, they could miss the global optimum in the presence of disjoint safe regions in the policy space (see Fig. 1). Disjoint safe regions appear when learning an impedance controller for a robot arm, as we show in our experiments and in many other applications [8,20,21]. To address this limitation [21] proposes GOSAFE, which can provably and safely discover the safe global optimum in the presence of disjoint safe regions under mild conditions. To achieve this, it learns safe backup policies for different states and uses them to preserve safety when evaluating policies outside of the safe set. Specifically, it switches between actively exploring local safe regions in the state and policy space and safe global exploration. However, the active exploration in the state and policy space requires a coarse discretization of the space and is infeasible for all but the simplest systems with low-dimensional state spaces, [22] argues that dimension $d > 3$ is already challenging. As a result, GOSAFE cannot only handle most real-world dynamical systems, and is restricted to impractical systems with low-dimensional state spaces. The concept of switching between two exploration stages is also pursued in the stagewise safe optimization algorithm proposed in [23]. However, also [23] is restricted to an optimum connected to a safe initialization. Lastly, the general idea of learning backup policies is related to safety filters and control barrier functions [24–26]. Nevertheless, those methods require either availability or learning of a dynamics model besides learning the policy and are, therefore, model-based. In this work, we focus on a model-free approach.

### *1.2. Contributions*

This work presents GOSAFEOPT, the first model-free algorithm that can globally search optimal policies for safety-critical, real-world dynamical systems, i.e., systems with high-dimensional state spaces. GOSAFEOPT does not discretize and actively explores the state space. Therefore, it overcomes the main shortcomings and restrictions of GOSAFE, while still performing safe global exploration. This makes GOSAFEOPT the first and only model-free safe global exploration algorithm for real-world dynamical systems. Crucially, GOSAFEOPT leverages the *Markov property* of the system's state to learn backup policies which it uses to guarantee safety when evaluating policies outside the safe set. This novel mechanism for learning backup policies does not depend on the dimension of the state space. We provide high-probability safety guarantees for GOSAFEOPT and we prove that it recovers the safe globally optimal policy under assumptions that hold for many practical cases. Finally, we validate it in both simulated and real safety-critical path following experiments on a robotic arm (see Fig. 2), which is prohibitive for GOSAFE, the only competing model-free global safe search method. Further, we show that GOSAFEOPT achieves considerably better performance than SAFEOPT, a state-of-the-art method for local model-free safe policy search, and its high-dimensional variants. Table 1 compares GOSAFEOPT to SAFEOPT and GOSAFE in terms of safety guarantees, scalability, global exploration, and sample efficiency. It shows that GOSAFEOPT is the only method that can perform sample-efficient global exploration in high-dimensional systems while providing safety guarantees.

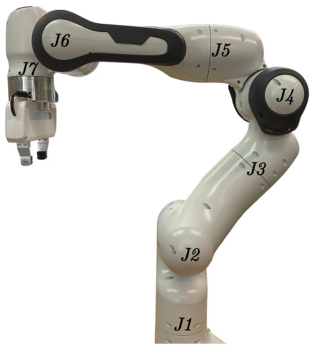

**Fig. 2.** Franka Emika Panda; seven degrees of freedom robot arm used for our evaluations.

**Table 1**
Comparison of GoSafeOpt and prior work on safe exploration based on their safety guarantees, scalability, global exploration, and sample efficiency.

| | Safe exploration | State space with dimension $d > 3$ | Global exploration | Sample efficient |
|---|---|---|---|---|
| **SafeOpt** [18] | ✓ | ✓ | ✗ | ✓ |
| **GoSafe** [21] | ✓ | ✗ | ✓ | ✗ |
| **GoSafeOpt (ours)** | ✓ | ✓ | ✓ | ✓ |

## 2. Problem setting

We consider a Lipschitz-continuous system

$$\mathrm{d}x(t) = z(x(t), u(t))\,\mathrm{d}t, \tag{1}$$

where $z(\cdot)$ represents the unknown system dynamics, $x(t) \in \mathcal{X} \subset \mathbb{R}^s$ is the system state and $u(t) \in \mathcal{U} \subset \mathbb{R}^p$ is the input we apply to steer the system state to follow a desired trajectory $x_{\text{des}}(t) \in \mathcal{X}$ for all $t \geq 0$. We assume that the system starts at a known initial state $x(0) = x_0$.

The control input $u(t)$ we apply for a given state $x(t)$ is specified by a policy $\pi : \mathcal{X} \times \mathcal{A} \to \mathcal{U}$, with $u(t) = \pi(x(t), a) := \pi^a(x(t))$. The policy is parameterized by $a \in \mathcal{A} \subset \mathbb{R}^d$, where $\mathcal{A}$ is a finite parameter space.[1] We encode our goal of following the desired trajectory $x_{\text{des}}(t)$ through an objective function, $f : \mathcal{A} \to \mathbb{R}$. Note, the trajectory of a deterministic system (1) is fully determined by its initial state $x_0$ and the control policy. Therefore, the objective is independent of the state space $\mathcal{X}$. We seek for a controller parametrization $a \in \mathcal{A}$ that optimizes $f$ for a constant initial condition $x_0$. Since the dynamics of the system in Eq. (1) is unknown, so is the objective $f$. Nonetheless, we assume we obtain a noisy measurement of $f(a)$ at any $a \in \mathcal{A}$ by running an experiment. We aim at optimizing $f$ from these measurements in a sample-efficient way. Additionally, to avoid the deployment of harmful policies, we formulate safety as a set of *unknown* constraints over the system trajectories that must be satisfied *at all times*. Similar, as for $f$, these constraints only depend on the parameter $a$ and hence take the form $g_i : \mathcal{A} \to \mathbb{R}$ for each constraint function $g_i$, where $i \in \{1, \dots, q\} := \mathcal{I}_g$ and $q \in \mathbb{N}$. The resulting constrained optimization problem with unknown objective and constraints is:

$$\max_{a \in \mathcal{A}} f(a) \quad \text{subject to} \quad g_i(a) \geq 0, \forall i \in \mathcal{I}_g. \tag{2}$$

We represent the objective and constraints using a scalar-valued function in a higher dimensional domain, proposed by [18]:

$$h(a, i) = \begin{cases} f(a) & \text{if } i = 0, \\ g_i(a) & \text{if } i \in \mathcal{I}_g, \end{cases} \tag{3}$$

with $\mathcal{I}_g = \{1, \dots, q\}$, $\mathcal{I} := \{0, 1, \dots, q\}$, and $i \in \mathcal{I}$. This representation will later help us in learning the unknown function.

In summary, *our goal is to find the optimal and safe policy parameter for the system starting from the nominal initial condition* $x_0$. We refer to the solution of Eq. (2) as the safe global optimum $a^*$. Note, finding the optimal policy for a fixed initial condition $x_0$ is a common task in episodic RL [1].

Solving this problem without a dynamics model and without incurring failures for generic systems, objectives, and constraints is hopeless. The following section introduces our assumptions to make this problem tractable.

[1] Infinite parameter spaces can be handled via discretization (e.g., random subsampling).

### 2.1. Assumptions

To solve the problem in Eq. (2) safely, we assume to have at least one initial safe policy to start data collection without violating constraints. This initial policy could be derived from available simulators, first principles models, or by performing controlled experiments on the hardware directly. This policy can be conservative and sub-optimal. For instance, for mobile robots, a policy that barely moves the robot could be an initial safe policy.

**Assumption 2.1.** A set $S_0 \subset \mathcal{A}$ of safe parameters is known. That is, for all parameters $a$ in $S_0$ we have $g_i(a) \geq 0$ for all $i \in \mathcal{I}_g$.

In practice, similar policies often lead to similar outcomes. In other words, the objective and the constraints exhibit regularity properties. We capture this by assuming that the function $h$, Eq. (3), lives in an reproducing kernel Hilbert space (RKHS) [27] and has bounded norm in that space.

**Assumption 2.2.** The function $h$ lies in an RKHS associated to a kernel $k$ and has a bounded norm in that RKHS $\|h\|_k \leq B$. Furthermore, the objective $f$ and constraints $g_i$ are Lipschitz continuous with known constants.

Without Assumption 2.2, the constraint and reward functions can be discontinuous making it impossible to infer the safety of a policy before evaluating it and to provide safety guarantees. In practical applications, such behavior is undesirable, and therefore rare. For further discussion on the practicality of this assumption, we refer the reader to [28].

Next, we formalize our assumptions on the measurement model.

**Assumption 2.3.** We obtain noisy measurements of $h$ with the measurement noise independent and identically distributed (i.i.d.) $\sigma$-sub-Gaussian. That is, for a measurement $y_i$ of $h(\cdot, i)$, we have $y_i = h(a, i) + \epsilon_i$ with $\epsilon_i$ $\sigma$-sub-Gaussian for all $i \in \mathcal{I}$.

Assumptions 2.1, 2.2, and 2.3 are common in the safe BO literature [16–18]. However, these approaches treat the evaluation of a policy as a black box. In contrast, we monitor the rollout of a policy to intervene and bring the system back to safety, if necessary. This can be achieved for a Markovian [29] system, like the one we consider in Eq. (1) (see Proposition A.3 in the appendix).

To monitor the rollouts, we assume that we receive a state measurement after every $\Delta t$ seconds and that in between discrete time steps, the system cannot arbitrarily jump, i.e., its movement within these (typically small) time intervals is bounded. Note, for many robotic systems this assumption is valid. Especially, since we can choose the sampling time $\Delta t$. However, estimating this bound can be challenging. A conservative value for the bound may be estimated by performing controlled experiments, e.g., with the safe initial policy from Assumption 2.1, directly on hardware. Simulators or first principle models, if available, can also be leveraged.

**Assumption 2.4.** The state $x(t)$ is measured after every $\Delta t$ seconds. Furthermore, for any $x(t)$ and $\rho \in [0, 1]$, the distance to $x(t + \rho\Delta t)$ induced by any action is bounded by a known constant $\Xi$, that is, $\|x(t + \rho\Delta t) - x(t)\| \leq \Xi$.

*Remark*: Implicitly, we here assume noise-free measurements of the state for simplicity. Our method also works for the noisy case (see Appendix A.1.1), which is typical in the real world.

Triggering a backup policy for a Markovian system is not sufficient to guarantee the safety of the whole trajectory for a generic constraint. Consider the case where safety is expressed as a constraint on a cost accumulated along the trajectory. Even if we are individually safe before and after triggering a backup policy, we might be unsafe overall. Therefore, we limit the types of constraints we consider.

**Assumption 2.5.** We assume that, for all $i \in \{1, \dots, q\}$, $g_i$ is defined as the minimum of a state-dependent function $\bar{g}_i$ along the trajectory starting in $x_0$ with controller $\pi^a$. Formally:

$$g_i(a) = \min_{x' \in \xi_{(0,x_0,a)}} \bar{g}_i(x'), \tag{4}$$

with $\xi_{(0,x_0,a)} := \{x_0 + \int_0^t z(x(\tau); \pi^a(x(\tau))\mathrm{d}\tau\}$ the trajectory of $x(t)$ under policy parameter $a$ starting from $x_0$ at time 0.

An example of such a constraint is the minimum distance of the system to an obstacle. We can now provide a formal definition of a safe experiment.

**Definition 2.6.** An experiment is safe if, for all $t \geq 0$ and all $i \in \{1, \dots, q\}$,

$$\bar{g}_i(x(t)) \geq 0. \tag{5}$$

This is a more general way of defining safety for the optimization problem from Eq. (2). In particular, where Eq. (2) only considers trajectories associated with a fixed policy parameter $a$, Definition 2.6 also covers the case in which different portions of the trajectory are induced by different controllers.

## 3. Preliminaries

This section reviews Gaussian processes (GPs) and how to use them to construct frequentist confidence intervals, as well as relevant prior work on safe exploration (SafeOpt).

### 3.1. Gaussian processes

We model our unknown objective and constraint functions using Gaussian process regression (GPR) [30]. In GPR, our prior belief is captured by a GP, which is fully determined by a prior mean function[2] and a covariance function $k\left(a, a'\right)$. Importantly, if the observations are corrupted by i.i.d. Gaussian noise with variance $\sigma^2$, i.e., $y_i = f(a_i) + v_i$, and $v_i \sim \mathcal{N}(0, \sigma^2)$, the posterior over $f$ is also a GP whose mean and variance can be computed in closed form. Let us denote with $Y_n \in \mathbb{R}^n$ the array containing $n$ noisy observations of $f$, then the posterior of $f$ at $\bar{a}$ is $f(\bar{a}) \sim \mathcal{N}\left(\mu_n(\bar{a}), \sigma_n^2(\bar{a})\right)$ where

$$\mu_n(\bar{a}) = k_n(\bar{a})(K_n + I_n\sigma^2)^{-1}Y_n, \tag{6a}$$

$$\sigma_n^2(\bar{a}) = k(\bar{a}, \bar{a}) - k_n(\bar{a})(K_n + I_n\sigma^2)^{-1}k_n^T(\bar{a}). \tag{6b}$$

The entry $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ of the covariance matrix $K_n \in \mathbb{R}^{n\times n}$ is $k\left(a_i, a_j\right)$, $k_n(\bar{a}) = [k(\bar{a}, a_1), \dots, k(\bar{a}, a_n)]$ captures the covariance between $x^*$ and the data, and $I_n$ is the $n \times n$ identity matrix.

Eq. (6) considers the case where $f$ is a scalar function. To model the objective $f$ *and* constraints $g_i$, we use the selector function from Eq. (3).

### 3.2. Frequentist confidence intervals

To avoid failures, we must determine the safety of a given policy before evaluating it. To this end, we reason about plausible worst-case values of the constraint $g_i$ for a new policy $a$. We use the posterior distribution over the objective and constraints given by Eq. (6) to build frequentist confidence intervals that hold with high probability, i.e., at least $1 - \delta$, and are of the form:

$$|\mu_{n-1}(a, i) - h(a, i)| \leq \beta_n^{1/2}\sigma_{n-1}(a, i), \quad \forall i \in \mathcal{I}. \tag{7}$$

For functions fulfilling Assumption 2.2 and 2.3, [31,32] derive an appropriate value for $\beta_n$. This value depends on $\delta$, $n$ and the maximum information gain $\gamma_n$, cf., [33].[3]

### 3.3. SafeOpt *for model-free safe exploration*

SafeOpt leverages the confidence intervals presented in Section 3.2 to solve black-box constrained optimization problems while guaranteeing safety for all the iterates with high probability. It ensures safety by limiting its evaluations to a set of provably safe inputs. In particular, SafeOpt defines the lower bound of the confidence interval $l_n$ as $l_n(a, i) = \max\{ l_{n-1}(a, i), \mu_{n-1}(a, i) - \beta_n^{1/2}\sigma_{n-1}(a, i)\}$, with $l_0(a, i) = 0$ for all $a \in S_0$, $i \in \mathcal{I}_g$ and $-\infty$ otherwise, and the upper bound $u_n$ as $u_n(a, i) = \min\{u_{n-1}(a, i), \mu_{n-1}(a, i) + \beta_n^{1/2}\sigma_{n-1}(a, i)\}$ with $u_0(a, i) = \infty$ for all $a \in \mathcal{A}, i \in \mathcal{I}$. Given a set of safe parameters $S_{n-1}$, it then infers the safety of nearby parameters by combining the confidence intervals with the Lipschitz continuity of the constraints:

$$S_n := \bigcap_{i\in\mathcal{I}_g} \bigcup_{a'\in S_{n-1}} \{a \in \mathcal{A} \mid l_n(a', i) - L_\mathrm{a} \left\|a - a'\right\| \geq 0\}, \tag{8}$$

with $L_a$ the joint Lipschitz constant of $f(a)$, $g_i(a)$. This leads to a local expansion of the safe set. Thus, in the case of disconnected safe regions, the optimum discovered by SafeOpt may be local (see Fig. 1).

[2] Assumed to be zero without loss of generality (w.l.o.g).

[3] The maximum information gain is $\gamma_n := \max_{A\subset D:|A|=n} I(y_A; f_A)$, where $I(y_A; f_A)$ is the mutual information between $f_A$ evaluated at points in $A$ and the observations $y_A$, that is the amount of information $y_A$ contains about $f$ [33].

**Algorithm 1** Local Safe Exploration (**LSE**).

```
Input: Safe set $S$, set of backups $\mathcal{B}$, dataset $\mathcal{D}$
1: Recommend parameter $a_n$ with Eq. (9)
2: Collect $\mathcal{R} = \bigcup_{k \in \mathbb{N}} \{a_n, x(k)\}$ and $h(a_n, i) + \varepsilon_n$
3: $\mathcal{B} = \mathcal{B} \cup \mathcal{R}$, $\mathcal{D} = \mathcal{D} \cup \{a_n, h(a_n, i) + \varepsilon_n\}$
4: Update sets $S$, $\mathcal{G}$, and $\mathcal{M}$          //Eq. (8), Appendix D Definitions D.1 and D.2
Return: $S$, $\mathcal{B}$, $\mathcal{D}$
```

## 4. GoSafeOpt

In this section, we present our algorithm, GoSafeOpt, which combines the sample efficient local exploration of SafeOpt with global exploration to safely discover globally optimal policies for dynamical systems. To the best of our knowledge, GoSafeOpt is the first model-free algorithm that can globally search for optimal policies, guarantee safety during exploration, and is applicable to complex hardware systems.

### 4.1. The algorithm

GoSafeOpt consists of two alternating stages, local safe exploration (**LSE**) and global exploration (**GE**). In **LSE**, we explore the safe portion of the parameter space connected to our current estimate of the safe set. Crucially, we exploit the Markov property to learn backup policies for each state we visit during **LSE** experiments. During **GE**, we evaluate potentially unsafe policies in the hope of identifying new, disconnected safe regions. The safety of this step is guaranteed by triggering the backup policies learned during **LSE** whenever necessary. If a new disconnected safe region is identified, we switch to a **LSE** step. Otherwise, GoSafeOpt terminates and recommends the optimum $a^* = \arg\max_{a \in S_n} l_n(a, 0)$.

In the following, we explain the **LSE** and **GE** stages in more detail and provide their pseudocode in Algorithms 1 and 2, respectively. Algorithm 4 presents the pseudocode for the full GoSafeOpt algorithm.

#### 4.1.1. Local safe exploration

Similar to SafeOpt, during **LSE** we restrict our evaluations to provably safe policies, i.e., policies in the safe set, which is initialized with the safe seed from Assumption 2.1 and is updated recursively according to Eq. (8) (line 4 in Algorithm 1). We focus our evaluations on two relevant subsets of the safe set introduced in [16]: the maximizers $\mathcal{M}_n$, i.e., plausibly optimal parameters, and the expanders $\mathcal{G}_n$, i.e., parameters that, if evaluated, could optimistically enlarge the safe set. For their formal definitions, see [16] or Appendix D. During **LSE**, we evaluate the most uncertain parameter, i.e., the parameter with the widest confidence interval, among the expanders and the maximizers:

$$a_n = \arg\max_{a \in \mathcal{G}_n \cup \mathcal{M}_n} \max_{i \in \mathcal{I}} w_n(a, i), \tag{9}$$

where $w_n(a, i) = u_n(a, i) - l_n(a, i)$.

As a by-product of these experiments, GoSafeOpt learns backup policies for all the states visited during these rollouts by leveraging the Markov property. Intuitively, for any state $x(t)$ visited when deploying a safe policy $a$ starting from $x_0$, we know that the sub-trajectory $\{x(\tau)\}_{\tau \geq t}$ is also safe because of Assumption 2.5. Moreover, this sub-trajectory is safe regardless of how we reach $x(t)$ since the state is Markovian. Thus, $a$ is a valid backup policy for $x(t)$.

This means we *learn about backup policies for multiple states during a single **LSE** experiment*. To make them available during **GE**, we introduce the set of backups $\mathcal{B}_n \subseteq \mathcal{A} \times \mathcal{X}$. After running an experiment with policy $a$, we collect all the discrete state measurements in the rollout $\mathcal{R} = \bigcup_{k \in \mathbb{N}} \{a, x(k)\}$ and add it to the set of backups, $\mathcal{B}_{n+1} = \mathcal{B}_n \cup \mathcal{R}$ (see Algorithm 1 line 3).

We perform **LSE** until the connected safe set is fully explored and the optimum within the safe set is discovered. Intuitively, this happens when we have learned our constraint and objective functions with high precision, i.e., when the uncertainty among the expanders and maximizers is less than $\epsilon$, and yet the safe set does not expand any further,

$$\max_{a \in \mathcal{G}_{n-1} \cup \mathcal{M}_{n-1}} \max_{i \in \mathcal{I}} w_{n-1}(a, i) < \epsilon \text{ and } S_{n-1} = S_n. \tag{10}$$

Note, GoSafeOpt, like SafeOpt, only explores the connected safe set in the parameter space and learns backup policies via the Markov property.

#### 4.1.2. Global exploration

**GE** aims at discovering new, disconnected safe regions. In particular, during a **GE** step, we evaluate the most uncertain parameter, i.e., with the highest value for $\max_{i \in \mathcal{I}_g} w_n(a, i)$, outside of the safe set, $a \in \mathcal{A} \setminus S_n$. As this parameter is not in our safe set, it is not guaranteed to be safe. Therefore, we monitor the state during the experiment and trigger a backup policy, learned during **LSE**, if we cannot guarantee staying in a safe region of the state space when continuing with the current choice of policy parameters (cf. Fig. 3).

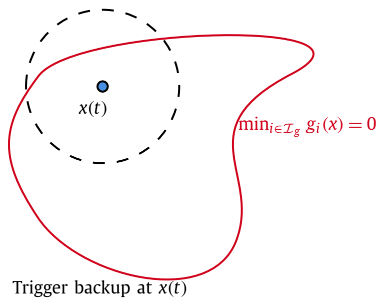

**Fig. 3.** Illustration of the boundary condition. *The backup policy is triggered at $x(t)$ if we cannot guarantee with high probability that all states in a ball around $x(t)$ are safe (see Assumption 2.4 and Section 4.1.2).*

**Algorithm 2** Global Exploration (**GE**).

```
Input: Safe set $S$, confidence intervals $C$, set of backups $\mathcal{B}$,
dataset $\mathcal{D}$, fail sets: $\mathcal{E}$, $\mathcal{X}_{\text{Fail}}$
1: Recommend global parameter $a_n$ with Eq. (11)
2: $a = a_n$, $x_{\text{Fail}} = \emptyset$, Boundary = False
3: while Experiment not finished do                                    //Rollout policy
4:     $x(k) = x_0 + \int_{t=0}^{kT} z\left(x(t), \pi\left(x(t); a\right)\right) dt$
5:     if Not Boundary then                                            //Not at boundary yet
6:         Boundary, $a_s^*$ = Boundary Condition($x(t)$, $\mathcal{B}$)
7:         if Boundary then                                            //Trigger backup policy
8:             $a = a_s^*$, $x_{\text{Fail}} = x(k)$
9:             $\mathcal{E} = \mathcal{E} \cup \{a_n\}$, $\mathcal{X}_{\text{Fail}} = \mathcal{X}_{\text{Fail}} \cup \{x_{\text{Fail}}\}$        //update fail sets
10: Collect $\mathcal{R} = \bigcup_{k\in\mathbb{N}} \{a_n, x(k)\}$, and $h(a_n, i) + \varepsilon_n$
11: if Not Boundary then                                               //Successful global search
12:     $\mathcal{B} = \mathcal{B} \cup \mathcal{R}$ and $\mathcal{D} = \mathcal{D} \cup \{a_n, h(a_n, i) + \varepsilon_n\}$
13:     $S = S \cup a$, $C(a, i) = C(a, i) \cap [0, \infty]$ for all $i \in \mathcal{I}_g$.
Return: $S$, $C$, $\mathcal{B}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{X}_{\text{Fail}}$
```

**Algorithm 3** Boundary Condition.

```
Input: $x$, $\mathcal{B}_n$
1: if $\forall (a_s, x_s) \in \mathcal{B}_n, \exists i \in \mathcal{I}_g, l_n(a_s, i) - L_x \left(\|x - x_s\| + \Xi\right) < 0$ then
2:     Boundary = True, Calculate $a_s^*$ (Eq. (12))
3: else
4:     Boundary = False, $a_s^* = \{\}$
return: Boundary, $a_s^*$
```

If a backup policy is triggered when evaluating the parameter $a$, we mark the experiment as failed. To avoid repeating the same experiment, we store $a$ and the state $x_{\text{Fail}}$ where we intervened in sets $\mathcal{E} \subset \mathcal{A}$ and $\mathcal{X}_{\text{Fail}} \subset \mathcal{X}$, respectively (see line 9 in Algorithm 2). Thus, during **GE**, we employ the following acquisition function

$$a_n = \underset{a\in\mathcal{A}\setminus(S_n\cup\mathcal{E})}{\arg\max} \; \max_{i\in\mathcal{I}_g} w_n(a, i). \tag{11}$$

This picks the most uncertain parameter, i.e., the parameter with the widest confidence interval, that is not provably safe but that has not been shown to trigger a backup policy. If the experiment was run without triggering a backup, we know that $a$ is safe. Therefore, we add the observed values for $g_i$ and $f$ to the dataset and the rollout $\mathcal{R}$ collected during the experiment to our set of backups $\mathcal{B}_n$, i.e., $\mathcal{B}_{n+1} = \mathcal{B}_n \cup \mathcal{R}$. Furthermore, we add the parameter $a$ to our safe set and update its lower bound, i.e., $l_n(a, i) = 0$, $\forall i \in \mathcal{I}_g$ (see lines 12 and 13 in Algorithm 2). Then, we switch to **LSE** to explore the newly discovered safe area. Note, the lower bound is updated again before the **LSE** step, i.e., $l_{n+1}(a, i) = \max\{l_n(a, i), \mu_n(a, i) - \beta_{n+1}^{1/2}\sigma_n(a, i)\}$ for all $i \in \mathcal{I}_g$ (see Algorithm 4 line 6).

If $\mathcal{A} \setminus (S_n \cup \mathcal{E}) = \emptyset$, there are no further safe areas we can discover and **GE** has converged.

### *4.1.3. Boundary condition*

Throughout each **GE** experiment, we monitor the state evolution, and, whenever a state measurement is received, we evaluate ***online*** a boundary condition to determine whether a backup policy should be triggered. Ideally, it must *(i)* guarantee safety, (*ii*) be fast to evaluate even for high-dimensional dynamical systems, and (*iii*) incorporate discrete-time measurements of the state. To fulfill requirement *(i)*, the boundary condition leverages Lipschitz continuity of the constraint.

**Algorithm 4** GoSafeOpt.

```
Input: Domain $\mathcal{A}$, $k(\cdot,\cdot)$, $S_0$, $C_0$, $\mathcal{D}_0$, $\kappa$, $\eta$
1: Initialize GP $h(a,i)$, $\mathcal{E}=\emptyset$, $\mathcal{X}_{\text{Fail}}=\emptyset$, $B_0=\{(a,x_0)\mid a\in S_0\}$
2: while $S_n$ expanding or $\mathcal{A}\setminus(S_n\cup\mathcal{E})\neq\emptyset$ do
3:     for $x\in\mathcal{X}_{\text{Fail}}$ do                                    //reevaluate fail sets
4:         if Not Boundary Condition$(x,\mathcal{B}_n)$ then                   //Algorithm 3
5:             $\mathcal{E}=\mathcal{E}\setminus\{a\}$, $\mathcal{X}_{\text{Fail}}=\mathcal{X}_{\text{Fail}}\setminus\{x\}$          //Update fail sets
6:     Update $C_n(a,i):=[l_n(a,i),u_n(a,i)]$ $\forall$ $a\in\mathcal{A}$, $i\in\mathcal{I}_g$            //see Section 3.3
7:     if LSE not converged (Eq. (10)) then                           //Perform LSE (Algorithm 1)
8:         $S_{n+1},\mathcal{B}_{n+1},\mathcal{D}_{n+1}=$ LSE$(\mathcal{S}_n,\mathcal{B}_n,\mathcal{D}_n)$
9:     else                                                          //Perform GE (Algorithm 2)
10:        $S_{n+1},C_{n+1},\mathcal{B}_{n+1},\mathcal{D}_{n+1},\mathcal{E},\mathcal{X}_{\text{Fail}}=$ GE$(\mathcal{S}_n,C_n,\mathcal{B}_n,\mathcal{D}_n,\mathcal{E},\mathcal{X}_{\text{Fail}})$
return: $\arg\max_{a\in S_n} l_n(a,0)$
```

In particular, when we are in $x(t)$, we check if there is a point $(a_s,x_s)$ in our set of backups $\mathcal{B}_n$ such that $x_s$ is sufficiently close to $x(t)$ to guarantee that $a_s$ can steer the system back to safety for any state we may reach in the next time step.

*Boundary Condition:* During iteration $n$, we trigger a backup policy at $x$ if there is no point in our set of backups $(a_s,x_s)\in\mathcal{B}_n$ such that $l_n(a_s,i)\geq L_{\mathrm{x}}\left(\|x-x_s\|+\Xi\right)$ for all $i\in\mathcal{I}_g$. In this case, we use the backup parameter $a_s^*$ with the highest safety margin, that is

$$a_s^*=\max_{\{a_s\in\mathcal{A}\mid\exists x_s\in\mathcal{X};(a_s,x_s)\in\mathcal{B}_n\}}\min_{i\in\mathcal{I}_g} l_n(a_s,i)-L_{\mathrm{x}}\|x-x_s\|\,. \tag{12}$$

Since we already calculate $l_n(a_s,i)$ for all $i\in\mathcal{I}_g$ and $a_s\in S_n$ *offline* to update the safe set (see Eq. (8)), we only need to evaluate $\|x-x_s\|$ *online*, which is computationally tractable for most real-world systems (e.g., $\mathcal{O}(s)$ for the 2-norm, where $s$ is the dimension of $\mathcal{X}$). Thereby, it satisfies requirement *(ii)* and enables the application of our algorithm to complex systems with high sampling frequencies. The boundary condition is summarized in Algorithm 3.

*Updating Fail Sets.* Parameters for which the boundary condition is triggered, i.e., parameters evaluated unsuccessfully during **GE**, are added to the fail set $\mathcal{E}$. However, when **LSE** is repeated after discovering a new region during **GE**, we can learn new backup policies, which makes the boundary condition less restrictive. Hence, it may happen that a parameter $a$ for which a backup policy was triggered during a previous **GE** step, i.e., $a\in\mathcal{E}$, we would not trigger a backup policy after **LSE** step has converged in the new safe region. Thus, after learning new backup policies during **LSE**, we re-evaluate the boundary condition (line 3), and update $\mathcal{E}$ and $\mathcal{X}_{\text{Fail}}$ accordingly. These states may then be revisited during further **GE** steps.

In summary, GoSafeOpt involves two alternating stages, **LSE** and **GE**. **LSE** steps are similar to SafeOpt, nonetheless, they additionally leverage the Markov property of the system to learn backup policies. These backup policies are then used in **GE** for global exploration. The only model-free safe exploration method that explores globally is GoSafe. However, it evaluates a completely different and expensive boundary condition, which relies on a safe set representation in the parameter and state space. This safe set is actively explored. Because of the active exploration, and expensive boundary condition, GoSafe becomes restricted to only systems with low-dimensional state spaces.

**Remark.** GoSafeOpt is devised for the episodic RL setting where the initial state $x_0$ is fixed and known. In several applications, the initial state is not known apriori and instead sampled i.i.d. from a state distribution $\rho$. Our formulation can also be extended to this setting by treating the initial state as a context variable, cf. [34]. Moreover, to guarantee safety in this setting, Assumption 2.1 has to be modified such that the parameters in the initial safe seed $S_0$, are safe for all initial states in the support of $\rho$, i.e., $x_0'\in\text{supp}(\rho)$. Then, given a context/initial state $x_0'$, the acquisition function for **LSE** or **GE** is optimized for the context. This is similar to the contextual SafeOpt algorithm [18]. The boundary condition can also be extended to incorporate the context. Finally, for a continuous state space, $\text{supp}(\rho)$ can be discretized similarly to as in GoSafe.

## 4.2. Theoretical results

This section provides safety (Section 4.2.1) and optimality (Section 4.2.2) guarantees for GoSafeOpt.

### 4.2.1. Safety guarantees

The main safety result for our algorithm is that GoSafeOpt guarantees safety during all experiments.

**Theorem 4.1.** *Under Assumptions 2.1 – 2.5 and with $\beta_n$ as defined in [18]. GoSafeOpt guarantees, for all $n\geq 0$ and any $\delta\in(0,1)$, that experiments are safe as per Definition 2.6 with probability at least $1-\delta$.*

The proof of this theorem is provided in Appendix A.1. Intuitively, we can analyze the safety of **LSE** and **GE** separately. For **LSE**, we can leverage the results in [18], which studies it extensively. Therefore, novel to our analysis is the safety of **GE**.

We show that while running experiments during **GE**, we can guarantee that if our boundary condition triggers a backup, we are safe, and if a backup is not triggered, then the experiment is safe, i.e., we discovered a new safe parameter.

#### *4.2.2. Optimality guarantees*

Next, we analyze when GoSafeOpt can find the safe global optimum $a^*$, which is the solution to Eq. (2). During **LSE**, we explore the connected safe region. For each safe region we explore, we can leverage the results from [18] to prove local optimality. Furthermore, due to **GE**, we can discover disconnected safe regions and then repeat **LSE** to explore them. To this end, we define when a parameter $a$ can be discovered by GoSafeOpt (either during **LSE** or during **GE**).

**Definition 4.2.** The parameter $a \in \mathcal{A}$ is discoverable by GoSafeOpt at iteration $n$, if there exists a set $A \subseteq S_n$ such that $a \in \bar{R}^c_\epsilon(A)$. Here, $\bar{R}^c_\epsilon(A)$ is the largest safe set we can safely reach from $A$ (see Eq. (A.13) in Appendix A.2 or [18,21]).

Next, we show that if the safe global optimum (solution of Eq. (2)) is discoverable as per Definition 4.2, then we can approximate it with $\epsilon$-precision.

**Theorem 4.3.** *Let $a^*$ be a safe global optimum. Further, let Assumptions 2.1 – 2.5 hold, $\beta_n$ be defined as in [18]. Assume there exists a finite integer $\tilde{n} \geq 0$ such that $a^*$ is discoverable at iteration $\tilde{n}$ (see Definition 4.2). Then, for any $\epsilon > 0$, and $\delta \in (0,1)$, there exists a finite integer $n^* \geq \tilde{n}$ such that with probability at least $1-\delta$,*

$$f(\hat{a}_n) \geq f(a^*) - \epsilon, \quad \forall n \geq n^* \tag{13}$$

*with $\hat{a}_n = \arg\max_{a \in S_n} l_n(a, 0)$.*

In practice, GoSafeOpt tends to find better controllers than SafeOpt, which converges after **LSE**. This is formalized in the following proposition.

**Proposition 4.4.** *For SafeOpt, $a^*$ is discoverable at iteration $n > 0$, if and only if, it is discoverable at iteration $n = 0$.*

Proposition 4.4 states that if the parameter $a^*$ does not lie in the largest safe set reachable from $S_0$, SafeOpt will not find it. GoSafeOpt does not suffer from the same restriction because of global exploration. In Appendix A.2.2, Lemma A.18 we provide additional conditions under which GoSafeOpt can find the safe global optimum. The performance benefits for GoSafeOpt are then empirically shown in Section 5.

**Remark.** The safety threshold $\delta$ is used to pick the designer's appetite for unsafe evaluations. For a large value of $\delta$, more parameters are available for sampling at each iteration. Accordingly, the method converges faster, however, while also allowing more unsafe evaluations, see [18] for more detail.

### *4.3. Practical modifications*

In practice, we can further improve the sample and computational efficiency by introducing minor modifications. While they do not guarantee optimality, they yield good results for our evaluation in Section 5. Furthermore, all the proposed modifications *do not affect the safety guarantees* of the method, and thus can be safely applied in practice.

#### *4.3.1. Fixing iterations for each stage*

In Algorithm 4, we perform a global search, i.e., **GE**, after the convergence of **LSE**. Nonetheless, it may be beneficial to run **LSE** for a fixed amount of steps and then switch to **GE**, before **LSE**'s convergence. This heuristic allows for the early discovery of disconnected safe regions, which may improve sample efficiency. Moreover, this allows "jumping" between different safe regions of the domain that, even though would be connected if we ran the current **LSE** to convergence, are currently disconnected. To this end, we apply the following heuristic scheme: ($i$) run **LSE** for $n_{\mathrm{LSE}}$ steps, ($ii$) run **GE** for $n_{\mathrm{GE}}$ steps or until we have discovered new safe parameters, and ($iii$) if **GE** discovers a new region, return to ($i$). Else, return to ($i$) after **GE** completion, but with reduced $n_{\mathrm{LSE}}$. Note, the proposed scheme still retains optimality because we do not restrict the total number of iterations with the system. However, in practice, we additionally impose an upper bound on the interactions, and therefore $n_{\mathrm{LSE}}$, and $n_{\mathrm{GE}}$ influence the budget of global and local exploration, this affects optimality (cf., Appendix C).

#### *4.3.2. Updated boundary condition*

If required, the boundary condition can be further modified to reduce computation time by considering only a subset of the states collected from experiments. The updated boundary condition reduces the online computation time at the expense of a more conservative boundary condition. Due to this conservatism, we lose our optimality guarantees. In practice, however, we still achieve good results (see Section 5).

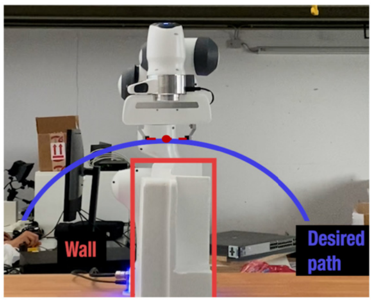

**Fig. 4.** Setup for our evaluation in Section 5. *We consider a safety-critical path following problem where deviations from the desired path (blue) could cause the robot to hit the wall (red box) and incur damage.*

**Definition 4.5.** Consider $\eta_l \in \mathbb{R}$ and $\eta_u \in \mathbb{R}$ such that $\eta_l < \eta_u$. The interior set $\Omega_{I,n}$ and marginal set $\Omega_{M,n}$ are defined as

$$\Omega_{I,n} = \{x_s \in \mathcal{X} \mid (a, x_s) \in \mathcal{B}_n : \forall i \in \mathcal{I}_g, l_n(a, i) \geq \eta_u\}$$
$$\Omega_{M,n} = \{x_s \in \mathcal{X} \mid (a, x_s) \in \mathcal{B}_n : \forall i \in \mathcal{I}_g, \eta_l \leq l_n(a, i) < \eta_u\}.$$

The interior set contains the points in our set of backups $\mathcal{B}_n$ that are safe with high tolerance $\eta_u$, whereas the points in the marginal set are safe with a smaller tolerance $\eta_l$. We use those sets for the updated boundary condition.

*Updated Boundary Condition*: Consider $d_l \in \mathbb{R}$ and $d_u \in \mathbb{R}$ such that $d_l < d_u$. We trigger a backup policy at $x$ if there is not a point $x_s \in \Omega_{I,n}$ such that $\|x - x_s\| \leq d_u$ or there is not a point $x'_s \in \Omega_{M,n}$ such that $\|x - x'_s\| \leq d_l$. In this case, we use the backup parameter $a^*_s$

$$a^*_s = \max_{\{a_s \in \mathcal{A} \mid (a_s, x_s) \in \mathcal{B}_n\}} l_n(a_s, i); \quad \text{with } x_s = \min_{x' \in \Omega_{I,n} \cup \Omega_{M,n}} \|x - x'\|. \tag{14}$$

Intuitively, we define distance tolerances $d_u$, and $d_l$ for points in $\mathcal{B}_n$ based on their safety tolerances $\eta_u, \eta_l$. As for Theorem 4.1, we can derive appropriate values for $\eta_u, d_u$, respectively $\eta_l, d_l$ to guarantee safety.

## 5. Evaluation

We evaluate GoSafeOpt in simulated and real experiments on a Franka Emika Panda seven degree of freedom (DOF) robot arm[4] (see Fig. 2 and 4). The objective of our experiments is to demonstrate that GoSafeOpt *(i)* can be applied to systems with high dimensional state spaces, *(ii)* is successful in safely tuning control parameters in common robotic tasks such as path following with manipulators, and *(iii)* is superior to the existing state-of-the-art method, SafeOpt, for safe control parameter tuning of real-world robotic systems.

Accordingly, in our results, we show that GoSafeOpt can scale to high dimensional systems, jump to disconnected safe regions while guaranteeing safety, and is directly applicable to hardware tasks with high sampling frequencies. In this work, we do not consider very high dimensional parameter spaces, which are in themselves challenging to tackle for methods such as SafeOpt. Methods in [35] and [22] alleviate this challenge and can be integrated with our algorithm easily. Thus, we concentrate on the novelty of our method, which is its globally safe parameter exploration, unlike SafeOpt, and scalability to high-dimensional state spaces compared to GoSafe. Specifically, the state space of systems we consider in this section is too large for GoSafe and it cannot be applied to any of our problems.

Details on the objective and constraint functions are provided in Appendix B. The hyperparameters of our experiments are listed in Appendix E.

In all experiments with the robot arm, we solely control the position and velocity of the end-effector. To this end, we consider an operational space impedance controller [36] with impedance gain $K$ (see Appendix B). The state space for our problem is six-dimensional. This is prohibitively large for GoSafe (struggles with state space greater than three [22]). Therefore, we compare our method with SafeOpt.

Impedance controllers for manipulators are usually tuned manually. This is often a tedious and time-consuming process. Accordingly, we show in our results that GoSafeOpt can be used to automate this tuning safely.

### 5.1. Simulation results

We first evaluate GoSafeOpt in a simulation environment based on the Mujoco physics engine [37].[5] For this we consider two distinct tasks, (*i*) reaching a desired position, and (*ii*) path following. We determine the impedance gain through an

[4] A video of our hardware experiments and link to code are available: https://sukhijab.github.io/GoSafeOpt/main_project.html.

[5] The URDFs and meshes are taken from https://github.com/StanfordASL/PandaRobot.jl.

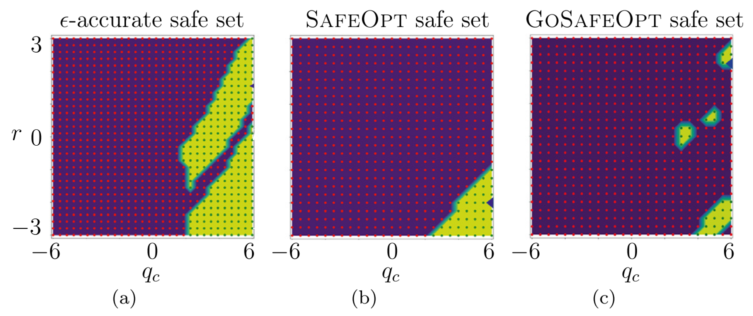

**Fig. 5.** Comparison of the safe set for simulation task between SafeOpt and GoSafeOpt after 200 iterations. *The yellow regions represent the safe sets. In each figure, the optimum is represented by a blue triangle.*

approximate model of the system and perform feedback linearization [36]. For the resulting linear system, we design a linear-quadratic regulator (LQR) [38] with quadratic costs that are parameterized by matrices $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{p\times p}$. Since the model is inaccurate, the feedback linearization will not cancel all nonlinearities and the LQR will not be optimal. Thus, our goal is to tune the cost matrices $Q$ and $R$ to compensate for the model mismatch. This approach is similar to [9]. We evaluate our methods over twenty independent runs of 200 iterations.

### 5.1.1. *Task 1: reaching a desired position*

We select a target $x_{\text{des}} \in \mathbb{R}^3$ for the robot. For this task, we parameterize the matrices $Q$ and $R$ by two parameters $(q_c, r) \in [2, 6] \times [-3, 3]$, that trade-off accurate tracking, i.e., large $q_c$, and small inputs, i.e., large $r$. We choose the objective function to encourage reaching the target as fast as possible while penalizing large end-effector velocities and control actions (see Appendix B.1 for details). Thus in total, we have an eight-dimensional task (six-dimensional state space and two-dimensional parameter space). For analysis purposes, we run a simple grid search, that we could not run outside of simulation, to get an estimate of the safe set and the global optimum. Fig. 5 depicts the $\epsilon$-precise ($\epsilon = 0.1$) safe set observed via grid search. From the figure, we observe that there is a disconnected safe region.

*Evaluation:* Fig. 5 depicts the safe sets of SafeOpt and GoSafeOpt after 200 learning iterations. We see that SafeOpt cannot discover the disconnected safe region and hence is stuck at a local optimum. On the other hand, GoSafeOpt discovers the disconnected regions and can jump within connected safe sets. The learning curve of the two methods is depicted in Fig. 7. Our method performs considerably better than SafeOpt. The optimum found by our method is 0.007 (less than $\epsilon = 0.1$) close to the optimum found via the grid search. SafeOpt cannot significantly improve over the initial policy. This is because the initial safe seed $S_0$ already contains a near-optimal policy from the connected region SafeOpt explores, i.e., $\max_{a\in S_0} f(a) \approx \max_{a\in \bar{R}^c_\epsilon(S_0)} f(a)$. Lastly, our method also achieves comparable safety to SafeOpt (on average 99.9% compared to 100%). We encounter the failures during **LSE**, which corresponds to SafeOpt, one could also expect similar behavior from SafeOpt if it were initialized in the upper region.

**Remark.** We can increase $\beta_n$ to encourage conservatism and avoid all unsafe evaluations. However, this also influences the algorithm's convergence rate. Hence, in practice, based on the task and appetite for unsafe evaluations $\beta_n$ has to be selected.

### 5.1.2. *Task 2: path following task*

For this experiment we define a parameterized path for the robot arm to follow $x_\text{d}(\rho(t))$. Here, we define $\rho(t)$ as a state to indicate progress along the trajectory, i.e., $x_\text{d}(0) = x_0$, $x_\text{d}(1) = x_{\text{des}}$. The evolution of $\rho(t) \in [0, 1]$ is controlled by a parameter $a_\rho \in [0, 1]$, that is, $\rho(t) = \min\{t(a_\rho\,(1/100 - 1/500) + 1/500), 1\}$. The objective is to find optimal control parameters for $Q$, $R$, and $a_\rho$ such that we progress on $x_\text{d}(\cdot)$ as fast as possible while ensuring that $\left\|x - x_\text{d}\big(\rho(t)\big)\right\|_2 \leq \zeta$. In this example, we model $Q$, $R$ using three parameters, $q_c, r, \kappa_d$, where $\kappa_d \in [0, 1]$ is used to weigh the velocity cost with respect to the positional cost of our state in the $Q$ matrix (cf. Appendix B.1). Together with $a_\rho$ as a parameter, this task is eleven-dimensional, with seven states (including $\rho$) and four parameters. This problem incorporates a challenging trade-off between fast trajectories and high-tracking performance. We compare it to SafeOptSwarm [35], a scalable version of SafeOpt for larger parameter spaces that use adaptive discretization. The results are presented in Fig. 8. Our results again show that GoSafeOpt performs considerably better than SafeOpt, specifically SafeOptSwarm. Furthermore, both SafeOpt and GoSafeOpt give 100% safety over all 20 runs. We also compare our method with expected improvement with constraints (EIC) [12] in Fig. 9. EIC discourages potentially unsafe regions but allows for unsafe evaluations. Our results show that EIC, and GoSafeOpt attain similar performance. However, EIC has considerably more unsafe evaluations (on average greater than fifteen) than GoSafeOpt, which has none.

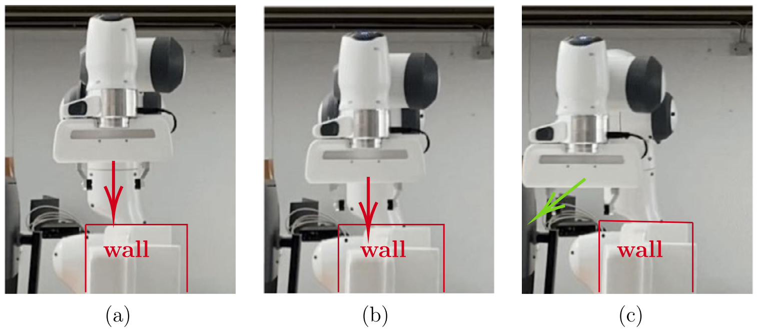

**Fig. 6.** Illustration of triggering the backup policy during **GE**. *During the global search, the policy directs the robot towards the wall in (a) and (b). A backup policy is automatically triggered by our boundary condition, once the robot gets too close to the wall. The backup policy directs the robot away from the wall (see green arrow in (c)).*

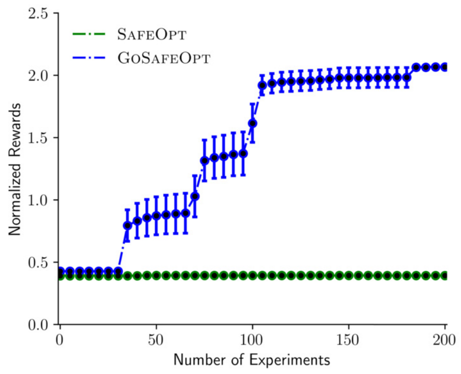

**Fig. 7.** Mean normalized objective with standard error for SafeOpt and GoSafeOpt for the eight-dimensional simulation task (20 runs).

### 5.2. Hardware results

While the simulation results already showcased the general applicability of GoSafeOpt to high dimensional systems and its ability to discover disconnected safe regions, we now demonstrate that it can also safely optimize policies on real-world systems.

*Control Task:* We consider a path following task (see the experimental setup in Fig. 4), and model the impedance gain $K$ as

$$K = \text{diag}\left(K_x, K_y, K_z, 2\sqrt{K_x}, 2\sqrt{K_y}, 2\sqrt{K_z}\right),$$

where $K_x = \alpha_x K_{r,x}$ with $K_{r,x} > 0$ a reference value used for Franka's impedance controller and $\alpha_x \in [0, 1.2]$ the parameter we would like to tune (same for $y, z$). Accordingly, $\alpha_{x,y,z} = 1$ corresponds to the impedance controller provided by the manufacturer. The parameter space we consider for this task is $[0, 1.2]^3$. We require the controller to follow the known desired path while avoiding the wall depicted in Fig. 4.

*Optimization Problem:* We choose our objective function to encourage tracking the desired path as accurately as possible and impose a constraint on the end-effector's distance from the wall (see Appendix B.2 for more details). We receive a measurement of the state at 250 Hz and evaluate the boundary condition during **GE** at 100 Hz.

*Evaluation:* The parameter space for this task is three-dimensional. Therefore, we compare our method to SafeOptSwarm [35] and run only 50 iterations for each algorithm in three independent runs. We choose $a_0 = (0.6, 0.6, 0.6)$ as our initial policy. During our experiments, both GoSafeOpt and SafeOptSwarm provide 100% safety in all three runs. For GoSafeOpt, safety during **GE** is preserved by triggering a backup policy if required. One such instance is shown in Fig. 6. We see in Fig. 10 that GoSafeOpt performs considerably better than SafeOptSwarm. In particular, even if we cannot prove the existence of disconnected safe regions for this task, GoSafeOpt still finds a better policy due to **GE**. Interestingly, the optimal value suggested by GoSafeOpt for both $\alpha_x$, and $\alpha_y$ is 1.2. Therefore, in the direction of our path, GoSafeOpt suggests aggressive controls to reduce tracking error. Moreover, the controller suggested by GoSafeOpt is more aggressive than the manufacturers' reference controller ($\alpha_x = 1.0$, $\alpha_y = 1.0$), and tracks the trajectory better.

#### 5.2.1. Choosing hyperparameters

GoSafeOpt, like many safe exploration BO algorithms such as SafeOpt and GoSafe, makes assumptions on prior knowledge of the system (see Section 2.1). These assumptions are crucial for theoretical guarantees. In practice, they are

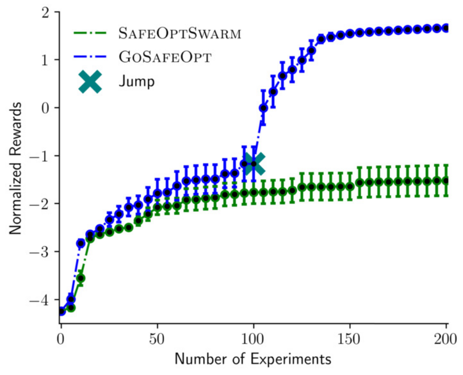

**Fig. 8.** Mean normalized objective with standard error for SafeOpt and GoSafeOpt for the eleven-dimensional simulation task (20 runs).

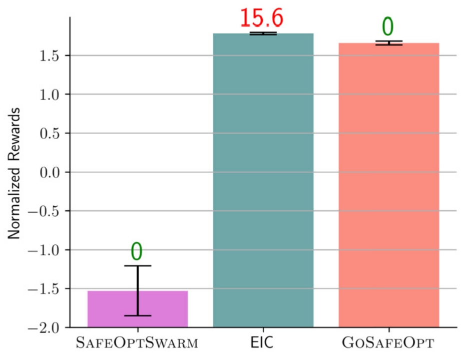

**Fig. 9.** Comparison of the normalized rewards with standard error and number of unsafe evaluations (numbers on top of the bars) between SafeOptSwarm, EIC, and GoSafeOpt for the eleven-dimensional simulation task (20 runs).

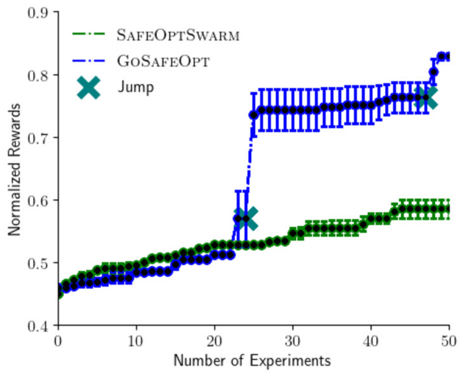

**Fig. 10.** Mean normalized objective with standard error for SafeOptSwarm and GoSafeOpt for the hardware task (3 runs). *The approximate location of the jump during **GE** is visible and indicated with a cyan cross.*

hard to verify. Yet, safe exploration BO methods have been successfully and safely applied to a large breath of applications [23,39–42]. In our case, we leverage the available simulator to obtain a range for the hyperparameters: kernel parameters, $\beta_n$, and distance metric for the boundary condition. Lastly, with $\beta_n$ fixed, we fine-tune the remaining parameters by performing controlled safe experiments with the hardware. Even though this approach gives good results, recent work from [43] investigates the hyperparameter selection problem for safe BO more systematically. In general, there are a few other works which investigate the gap between theory and practice [28,44]. Nonetheless, given the potential of these algorithms for reliable and safe artificial intelligence (AI), we acknowledge that future research on bridging this gap is needed.

## 6. Conclusion

This work proposes GoSafeOpt, a novel model-free learning algorithm for global safe optimization of policies for complex dynamical systems with high-dimensional state spaces. We provide for GoSafeOpt high probability safety guarantees and show that it provably performs better than SafeOpt, a state-of-the-art model-free safe exploration algorithm. We demonstrate the superiority of our algorithm over SafeOpt empirically through our experiments. GoSafeOpt can handle more complex and realistic dynamical systems compared to existing model-free learning methods for safe global exploration, such as GoSafe. This is due to a combination of an efficient passive discovery of backup policies that leverages the Markov property of the system and a novel and efficient boundary condition to detect when to trigger a backup policy. Future extensions could design hybrid algorithms that leverage the Markov property *and* actively explore the state space. Moreover, GoSafeOpt is designed for efficient and safe controller tuning. We believe it can be applied to other dynamical systems, e.g., in legged robotics, where controller parameter tuning is a crucial component [45].

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article.

## Acknowledgements

We would like to thank Kyrylo Sovailo for helping us with the hardware experiments on the Franka Emika Panda arm and Alonso Marco for insightful discussions and for providing the EIC code. Furthermore, we would also like to thank Christian Fiedler, Pierre-François Massiani, and Steve Heim for their feedback on this work.

This project has received funding from the Federal Ministry of Education and Research (BMBF) and the Ministry of Culture and Science of the German State of North Rhine-Westphalia (MKW) (grant number: StUpPD_407-21) under the Excellence Strategy of the Federal Government and the Länder, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program grant agreement No 815943, the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40 180545, and the Microsoft Swiss Joint Research Center (grant number 2019-037).

## Appendix A. Proofs of theoretical results

In this section, we provide proof for the theoretical results stated in the main body of the paper. In the following, we denote by $k$ discrete time indices and with $t$ continuous ones. This difference is important because, while we obtain state measurements at discrete times, we need to preserve safety at all times. Moreover, similarly to the notation in GoSafe [21], we denote by $\xi_{(t,x(t),a)} = \{x(t) + \int_t^{t'} z(x(\tau); \pi^a(x(\tau)))\mathrm{d}\tau \mid t' \geq t\}$ all the states in the trajectory induced by the policy $a$ starting from $x(t)$ at time $t$.

### A.1. Safety guarantees

In the following, we prove Theorem 4.1, which gives the safety guarantees for GoSafeOpt. Since GoSafeOpt has two stages, **LSE** and **GE**, we can study their safety separately. For **LSE**, [18] provides safety guarantees. Therefore, here we focus on the safety guarantees for **GE** and then show that combining both will guarantee the safety of the overall algorithm. To this end, we first make a hypothesis on our safe set $S_n$ and confidence bounds $l_n(a,i)$ and $u_n(a,i)$.

**Hypothesis A.1.** *Let $S_n \neq \emptyset$. The following properties hold for all $i \in \mathcal{I}_g$, $n \geq 0$ with probability at least $1-\delta$:*

$$\forall a \in S_n : g_i(a, x_0) \geq 0, \tag{A.1}$$

$$\forall a \in \mathcal{A} : l_n(a,i) \leq g_i(a, x_0) \leq u_n(a,i). \tag{A.2}$$

We leverage this hypothesis to prove that we are safe during **GE** and then we show that it is satisfied for GoSafeOpt. Particularly, during **LSE**, [18] proves that our hypothesis is fulfilled. Hence, before **GE**, the safe set and the confidence intervals satisfy it. In the following, we show the updates of the safe sets and the confidence intervals implemented by **GE** also satisfy our hypothesis, which is sufficient to conclude that the hypothesis is satisfied for all $n \geq 0$ (we will make this concrete in Lemma A.9).

During **GE**, we receive measurements of the state in discrete times and evaluate our boundary condition to trigger a backup policy if necessary. Therefore, we first show that even with discrete-time measurements, we can still guarantee safety in continuous time.

**Lemma A.2.** *Let Assumptions 2.4 and 2.5 hold and let $k_+ \geq k_- \geq 0$ be arbitrary integers. If, for all integers $k \in [k_-, k_+]$, there exists $a_s \in \mathcal{A}$ such that $g_i(a_s, x(k)) \geq L_x \Xi$ for all $i \in \mathcal{I}_g$, then $\bar{g}_i(x(t)) \geq 0$, for all $t \in [k_- \Delta t, (k_+ + 1)\Delta t]$ and $i \in \mathcal{I}_g$.*

**Proof.** By choice of the sampling scheme, we have that the state $x(k)$ measured in discrete time, corresponds to the state $x(k\Delta t)$ in continuous time. Hence, $g_i(a_s, x(k)) = g_i(a_s, x(k\Delta t))$. Consider some $k \geq 0$ and $a_s \in \mathcal{A}$ such that $g_i(a_s, x(k\Delta t)) \geq L_x \Xi$. For any $t \in [k\Delta t, (k+1)\Delta t]$ we have

$$\begin{aligned} g_i(a_s, x(k\Delta t)) - g_i(a_s, x(t)) &\leq L_x \|x(k\Delta t) - x(t)\| && \text{(Lipschitz continuity (Assumption 2.2))} \\ &\leq L_x \Xi. && \text{(Assumption 2.4)} \end{aligned}$$

Now, since $g_i(a_s, x(k\Delta t)) \geq L_x \Xi$, we have, for all $t \in [k\Delta t, (k+1)\Delta t]$ and $i \in \mathcal{I}_g$,

$$g_i(a_s, x(t)) \geq g_i(a_s, x(k\Delta t)) - L_x \Xi \geq 0. \tag{A.3}$$

For our choice of constraints (Assumption 2.5) this implies $\bar{g}_i(x(t)) \geq 0$ for all $i \in \mathcal{I}_g$ and $t \in [k\Delta t, (k+1)\Delta t]$. Finally, since this holds for all integers $k$ with $k_- \leq k \leq k_+$, it also holds for all $t \in [k_- \Delta t, (k_+ + 1)\Delta t]$. □

Now we have established a condition that guarantees for a given time interval that $\bar{g}_i(x(t)) \geq 0$ for all $i \in \mathcal{I}_g$.

We collect parameter and state combinations during rollouts in our set of backups $\mathcal{B}_n$. The intuition here is that for a Markovian system, all states visited during a safe experiment are also safe. This is important as it allows GoSafeOpt to learn backup policies for multiple states without actively exploring the state space. We formalize this in the following proposition.

**Proposition A.3.** *Let Assumption 2.5 hold. If $(a, x_0)$ is safe, that is, $\min_{x' \in \xi_{(0,x_0,a)}} \bar{g}_i(x') \geq 0$ for all $i \in \mathcal{I}_g$, then, for all $t_1 \geq 0$, $(a, x(t_1))$ is also safe, that is $\min_{x' \in \xi_{(t_1, x(t_1), a)}} \bar{g}_i(x') \geq 0$ for all $i \in \mathcal{I}_g$.*

**Proof.** The system in Eq. (1) is Markovian, i.e., for any $x(t_1) \in \xi_{(0,x_0,a)}$ and $x(t_2) \in \xi_{(0,x_0,a)}$ with $t_2 > t_1 > 0$,

$$\begin{aligned} x(t_2) &= x_0 + \int_0^{t_1} z(x(t); \pi^a(x(t))\mathrm{d}t + \int_{t_1}^{t_2} z(x(t); \pi^a(x(t))\mathrm{d}t \\ &= x(t_1) + \int_{t_1}^{t_2} z(x(t); \pi^a(x(t))\mathrm{d}t. \end{aligned}$$

Therefore, a trajectory starting in $x(t_1)$ will always result in the same state evolution, independent of how we arrived at $x(t_1)$. Combining this and Assumption 2.5, we get

$$\begin{aligned} g_i(a, x(t_1)) &= \min_{x' \in \xi_{(t_1, x(t_1), a)}} \bar{g}_i(x') && \text{Assumption 2.5} \\ &\geq \min_{x' \in \xi_{(0,x_0,a)}} \bar{g}_i(x') && \text{Markov Property} \\ &= g_i(a, x_0) && \text{Assumption 2.5} \\ &\geq 0. \quad \square \end{aligned}$$

In the following, we show that $g_i(a_s, x_0)$ is a lower bound for all points $(a_s, x_s)$ in $\mathcal{B}_n$, i.e., $g_i(a_s, x_s) \geq g_i(a_s, x_0)$. This will play a crucial role in showing that we preserve safety whenever we trigger a backup policy.

**Corollary A.4.** *Let Assumption 2.5 hold. For all points $(a_s, x_s)$ in $\mathcal{B}_n$, $g_i(a_s, x_s) \geq g_i(a_s, x_0)$ for all $i \in \mathcal{I}_g$.*

**Proof.** Each point $(a_s, x_s)$ in $\mathcal{B}_n$ is collected during a safe experiments (see Algorithm 1 line 3 and Algorithm 2 line 12). Therefore, $x_s \in \xi_{(0,x_0,a_s)}$. The result then follows from Proposition A.3. □

Corollary A.4 shows that $l_n(a_s, i)$ is a conservative lower bound on $g_i(a_s, x_s)$. Crucially, if we can observe not just the rollouts but also the constraint values $g_i(a_s, x_s)$, we could model them with a GP to obtain a potentially less conservative lower bound. However, in our work, we only assume that we can measure $g_i(a_s, x_0)$ (Assumption 2.3).

Proposition A.3 and Corollary A.4 formalize how we collect our backup policies and leverage them in our boundary condition. In the following, we prove that experiments, where we trigger a backup policy, are safe. First, we show that if the boundary condition is triggered at a time step $k^*$, then we are safe up until $k^*\Delta t$, i.e., time of trigger.

**Lemma A.5.** *Let the assumptions from Theorem 4.1 and Hypothesis A.1 hold. If, during **GE**, the boundary condition from Algorithm 3 triggers a backup policy at time step $k^* > 0$, then, for all $t \le k^*\Delta t$ and $i \in \mathcal{I}_g$, $\bar{g}_i(x(t)) \ge 0$ with probability at least $1 - \delta$.*

**Proof.** Consider $k < k^*$. Since the boundary condition (Algorithm 3) did not trigger a backup policy at $k$, we have

$$\exists (a_s, x_s) \in \mathcal{B}_n \text{ such that } l_n(a_s, i) \ge L_{\mathrm{x}} \left( \|x(k) - x_s\| + \Xi \right), \forall i \in \mathcal{I}_g. \tag{A.4}$$

By Lipschitz continuity of $g$, we have

$$g_i(a_s, x_s) - g_i(a_s, x(k)) \le L_{\mathrm{x}} \|x(k) - x_s\|, \tag{A.5}$$

which implies

$$\begin{aligned} g_i(a_s, x(k)) &\ge g_i(a_s, x_s) - L_{\mathrm{x}} \|x(k) - x_s\| \\ &\ge g_i(a_s, x_0) - L_{\mathrm{x}} \|x(k) - x_s\| && \text{Corollary A.4} \\ &\ge l_n(a_s, i) - L_{\mathrm{x}} \|x(k) - x_s\| && \text{Hypothesis A.1} \\ &\ge L_{\mathrm{x}} \Xi && \text{(A.4)} \end{aligned}$$

for all $i \in \mathcal{I}_g$ and $k < k^*$. Therefore, we can use Lemma A.2 to prove the claim by choosing $k_- = 0$ and $k_+ = k^* - 1$. □

Lemma A.5 shows that up until the time we trigger our boundary condition, we are safe with enough tolerance ($L_{\mathrm{x}}\Xi$) to guarantee safety. In the following, we show that if we trigger a safe backup policy at $k^*$, we will fulfill our constraints for all times after triggering.

**Lemma A.6.** *Let the assumptions from Theorem 4.1 and Hypothesis A.1 hold. If during a **GE** experiment with parameter $a_{\mathbf{GE}}$ at time step $k^* \ge 0$, our boundary condition triggers the backup policy $a_s^*$ defined as (see Eq. (12))*

$$a_s^* = \underset{\{a \in \mathcal{A} | \exists x \in \mathcal{X}; (a,x) \in \mathcal{B}_n\}}{\arg\max} \; \min_{i \in \mathcal{I}_g} l_n(a, i) - L_{\mathrm{x}} \left\| x - x(k^*) \right\|. \tag{A.6}$$

*Then for all $t \ge k^*\Delta t$, $\bar{g}_i(x(t)) \ge 0$ for all $i \in \mathcal{I}_g$ with probability at least $1 - \delta$.*

**Proof.** We want to show that Eq. (A.6) finds a parameter $a_s^*$ such that $g_i(a_s^*, x(k^*)) \ge 0$. For $k^* = 0$, this follows by definition because $\mathcal{B}_n$ consists of safe rollouts (see Algorithm 1 line 3 and Algorithm 2 line 12) and thus, for all parameters $a_s$ in $\mathcal{B}_n$ and $i \in \mathcal{I}_g$, we have $g_i(a_s, x_0) \ge 0$.

Let us now consider any integer $k^* > 0$. Let $(a_s, x_s) \in \mathcal{B}_n$ be arbitrary. Following the same Lipschitz continuity-based arguments as in Lemma A.5, we have for all $i \in \mathcal{I}_g$:

$$\begin{aligned} g_i(a_s, x(k^*)) &\ge l_n(a_s, i) - L_{\mathrm{x}} \left\| x_s - x(k^*) \right\| && \text{same as Lemma A.5} \\ &\ge l_n(a_s, i) - L_{\mathrm{x}} \Big( \left\| x(k^* - 1) - x_s \right\| + \left\| x(k^*) - x(k^* - 1) \right\| \Big) && \text{(Triangle inequality)} \\ &\ge l_n(a_s, i) - L_{\mathrm{x}} \left( \left\| x(k^* - 1) - x_s \right\| + \Xi \right) && \text{(Assumption 2.4)} \\ &\ge 0, \end{aligned} \tag{A.7}$$

where the last inequality follows from the fact that the boundary condition was not triggered at time step $k^* - 1$ (see Section 4.1.2). Furthermore, from Eq. (A.7) we can conclude that there exists $a_s \in \mathcal{A}$ such that for some $x_s \in \mathcal{X}$, $(a_s, x_s) \in \mathcal{B}_n$, and $l_n(a_s, i) - L_{\mathrm{x}} \|x_s - x(k^*)\| \ge 0$ for all $i \in \mathcal{I}_g$. Therefore, we have for $a_s^*$ recommended by Eq. (A.6):

$$\max_{\{a \in \mathcal{A} | \exists x \in \mathcal{X}; (a,x) \in \mathcal{B}_n\}} \; \min_{i \in \mathcal{I}_g} l_n(a, i) - L_{\mathrm{x}} \left\| x - x(k^*) \right\| \ge 0.$$

Hence, $g_i(a_s^*, x(k^*)) \ge 0$ for all $i \in \mathcal{I}_g$ with probability at least $1 - \delta$, which proves the claim. □

Lemmas A.5 and A.6 show that, if we trigger a backup policy during **GE**, we can guarantee the safety of the experiment before and after switching to the backup policy, respectively.

Next, we prove that, if the backup policy is not triggered during **GE** with parameter $a_{\mathbf{GE}}$, then $a_{\mathbf{GE}}$ is safe with high probability.

**Lemma A.7.** *Let the assumptions from Theorem 4.1 and Hypothesis A.1 hold. If, during* **GE** *with parameter* $a_{\mathbf{GE}}$*, a backup policy is not triggered by our boundary condition, then* $a_{\mathbf{GE}}$ *is safe with probability at least* $1-\delta$*, that is,* $g_i(a_{\mathbf{GE}}, x_0) \geq 0$ *for all* $i \in \mathcal{I}_g$*.*

**Proof.** Assume the experiment was not safe, i.e., there exists a $t \geq 0$, such that for some $i \in \mathcal{I}_g$ $\bar{g}_i(x(t)) < 0$. Consider the time step $k \geq 0$ such that $t \in [k\Delta t, (k+1)\Delta t]$. Since the boundary condition was not triggered during the whole experiment, it was also not triggered at time step $k$. This implies that (see Section 4.1.2) there exists a point $(a_s, x_s) \in \mathcal{B}_n$ such that

$$l_n(a_s, i) - L_x(\|x_s - x(k)\| + \Xi) \geq 0, \tag{A.8}$$

for all $i \in \mathcal{I}_g$. Therefore, we have $g_i(a_s, x(k)) \geq L_x\Xi$ (Hypothesis A.1). Hence, from Lemma A.2 we have $\bar{g}_i(x(t)) \geq 0$ for all $i \in \mathcal{I}_g$. This contradicts our assumption that for some $t \geq 0$ and $i \in \mathcal{I}_g$, $\bar{g}_i(x(t)) < 0$. □

The following Corollary summarizes the safety of **GE**.

**Corollary A.8.** *Under the assumptions from Theorem 4.1 and Hypothesis A.1* GoSafeOpt *is safe during* **GE***, i.e., for all* $t \geq 0$, $\bar{g}_i(x(t)) \geq 0$ *for all* $i \in \mathcal{I}_g$*.*

**Proof.** Two scenarios can occur during **GE**, $(i)$ a backup policy is triggered at some time step $k^* \geq 0$, $(ii)$ the experiment is completed without triggering a backup policy. For the first case, Lemma A.6 guarantees that we are safe after triggering the backup policy, and Lemma A.5 guarantees that we are safe before we trigger the backup. For second scenario, Lemma A.7 guarantees safety. □

We have now shown that under the assumptions of Theorem 4.1 combined with Hypothesis A.1, we can guarantee that we are safe during **GE**, irrespective of whether we trigger a backup policy or not. We leverage this result to show that Hypothesis A.1 is satisfied for GoSafeOpt.

**Lemma A.9.** *Let the assumptions from Theorem 4.1 hold and* $\beta_n$ *be defined as in [18]. Then, Hypothesis A.1 is satisfied for* GoSafeOpt*, that is, with probability at least* $1-\delta$ *for all* $i \in \mathcal{I}_g$ *and* $n \geq 0$

$$\forall a \in S_n : g_i(a, x_0) \geq 0, \tag{A.9}$$

$$\forall a \in \mathcal{A} : l_n(a, i) \leq g_i(a, x_0) \leq u_n(a, i). \tag{A.10}$$

**Proof.** We use induction on $n$.

*Base case* $n = 0$*:* By Assumption 2.1, we have, for all $a \in S_0$, $g_i(a, x_0) \geq 0$ for all $i \in \mathcal{I}_g$. Moreover, the initialization of the confidence intervals presented in Section 3.3 is as follows: $l_0(a, i) = 0$ if $a \in S_0$ and $-\infty$ otherwise, and $u_0(a, i) = \infty$ for all $a \in \mathcal{A}$. Thus, it follows that $l_0(a, i) \leq g_i(a, x_0) \leq u_0(a, i)$ for all $a \in \mathcal{A}$.
*Inductive step:* Our induction hypothesis is $l_{n-1}(a, i) \leq g_i(a, x_0) \leq u_{n-1}(a, i)$ and $g_i(a, x_0) \geq 0$ for all $a \in S_{n-1}$ and for all $i \in \mathcal{I}_g$. Based on this, we prove that these relations hold for iteration $n$.

We start by showing that $l_n(a, i) \leq g_i(a, x_0) \leq u_n(a, i)$ for all $a \in \mathcal{A}$. To this end, we distinguish between the different updates of the two stages of GoSafeOpt, **LSE** and **GE**. During **LSE**, we define $l_n(a, i)$ and $u_n(a, i)$ as

$$l_n(a, i) = \max(l_{n-1}(a, i), \mu_n(a, i) - \beta_n \sigma_n(a, i)),$$
$$u_n(a, i) = \min(u_{n-1}(a, i), \mu_n(a, i) + \beta_n \sigma_n(a, i)).$$

We know that $g_i(a, x_0) \geq l_{n-1}$ by induction hypothesis and $g_i(a, x_0) \geq \mu_n(a, i) - \beta_n \sigma_n(a, i)$ with probability $1-\delta$ from [18]. This implies $g_i(a, x_0) \geq l_n$. A similar argument holds for the upper bound.

During **GE**, we update $l_n(a, i)$ if the parameter we evaluate induces a trajectory that does not trigger a backup policy (see Algorithm 2 line 13). For this parameter, the induction hypothesis allows us to use Lemma A.7 and conclude $g_i(a, x_0) \geq 0$. Therefore, the update of the confidence intervals during **GE** also satisfies Eq. (A.10) for iteration $n$, thus completing the induction step for the confidence intervals.

As for the confidence intervals, we distinguish between the different updates of the safe set implemented by **LSE** and **GE**. In the case of **GE**, we update the safe set by adding the evaluated policy parameter $a$ only if it does not trigger a backup, i.e., $S_n = S_{n-1} \cup \{a\}$. Following the same argument as above, we can conclude $g_i(a, x_0) \geq 0$ for all $i \in \mathcal{I}_g$. This together with the induction hypothesis means $g_i(a, x_0) \geq 0$ for all $i \in \mathcal{I}_g$ and $a \in S_n$ in case of a **GE** update.

Now we focus on **LSE**. We showed Eq. (A.10) holds for $n$. Moreover, we know by induction hypothesis $g_i(a, x_0) \geq 0$ for all $a \in S_{n-1}$ and for all $i \in \mathcal{I}_g$ with high probability. The update equation for the safe set (Eq. (8)) gives for all $a' \in S_n \setminus S_{n-1}$, there exists $a \in S_{n-1}$ such that for all $i \in \mathcal{I}_g$

$$l_n(a, i) - L_{\text{a}} \left\| a - a' \right\| \geq 0. \tag{A.11}$$

We show that this is enough to guarantee with high probability that $g_i(a', x_0) \geq 0$. Due to the Lipschitz continuity of the constraint functions, we have

$$\begin{aligned} g_i(a', x_0) &\geq g_i(a, x_0) - L_{\text{a}} \left\| a - a' \right\|, \\ &\geq l_n(a, i) - L_{\text{a}} \left\| a - a' \right\| \geq 0. && \text{(Eq. (A.11))} \end{aligned}$$

Therefore, $g_i(a', x_0) \geq 0$ for all $i \in \mathcal{I}_g$ and $a \in S_n$ with probability at least $1 - \delta$ also in case of an **LSE** step. □

Lemma A.9 ensures that Hypothesis A.1 holds for GoSafeOpt. We also know that under the same assumption as Theorem 4.1 and Hypothesis A.1, we are safe during **GE** (see Corollary A.8). Hence, we can now guarantee safety during **GE**. Finally, we prove Theorem 4.1, which guarantees safety for GoSafeOpt.

**Theorem 4.1.** *Under Assumptions 2.1 – 2.5 and with $\beta_n$ as defined in [18].* GoSafeOpt *guarantees, for all $n \geq 0$ and any $\delta \in (0, 1)$, that experiments are safe as per Definition 2.6 with probability at least $1 - \delta$.*

**Proof.** We perform GoSafeOpt in two stages; **LSE** and **GE**. In Lemma A.9, we proved that for all parameters $a \in S_n$, $g_i(a, x_0) \geq 0$ for all $i \in \mathcal{I}_g$ with probability at least $1 - \delta$. During **LSE** we query parameters from $S_n$ (Eq. (9)). Therefore, the experiments are safe. During **GE**, Corollary A.8 proves that when assumptions from Theorem 4.1 and Hypothesis A.1 hold, we are safe during **GE** for our choice of $\beta_n$. Furthermore, in Lemma A.9 we proved that Hypothesis A.1 is satisfied for GoSafeOpt. Hence, we can conclude that if the assumptions from Theorem 4.1 hold, we are safe during **GE** at all times. □

### *A.1.1. Proof of boundary condition for noisy measurements*

**Lemma A.10.** *Assume at each time step $k$ we receive a noisy measurement of the state to evaluate our boundary condition, i.e., $y = x + \varepsilon$ and $\varepsilon$ i.i.d. Specifically, assume $P(\|\varepsilon\| \leq d/2) \geq \sqrt{1 - \delta_2}$. If we have for some $(a_s, y_s) \in \mathcal{B}_n$ $(y_s = x_s + \varepsilon_s)$*

$$l_n(a_s, i) - L_{\text{x}}(\|y - y_s\| + \Xi + d) \geq 0,$$

*then with probability at least $1 - \delta_2$ we have*

$$l_n(a_s, i) - L_{\text{x}}(\|x - x_s\| + \Xi) \geq 0.$$

**Proof.** We would like to show that

$$l_n(a_s, i) - L_{\text{x}}(\|y - y_s\| + \Xi + d) \leq l_n(a_s, i) - L_{\text{x}}(\|x - x_s\| + \Xi).$$

This implies that $d \geq \|x - x_s\| - \|y - y_s\|$.

Accordingly,

$$\begin{aligned} \|x - x_s\| - \|y - y_s\| &\leq \|x - x_s - (y - y_s)\| && \text{(reverse triangle inequality)} \\ &= \|\varepsilon_s - \varepsilon\| \leq \|\varepsilon\| + \|\varepsilon_s\| \\ &\leq d \quad \square && \text{(with probability at least } 1 - \delta_2.\text{)} \end{aligned}$$

Following the lemma, we can come up with a more conservative boundary condition (with one step jump bound $\Xi' = \Xi + d$) which still guarantees safety. However, the price we pay for not measuring our state perfectly is the additional probability term $1 - \delta_2$. Lastly, here we only look at the influence of noisy state measurements on the boundary condition. Nevertheless, if the policy $\pi$ uses some form of feedback, the noise also enters the dynamics. In this work, we assume that this influence is captured by our observation model, see Assumption 2.3.

## *A.2. Optimality guarantees*

In this section, we prove Theorem 4.3 which guarantees that the safe global optimum can be found with $\epsilon$-precision if it is discoverable at some iteration $n \geq 0$ (see Definition 4.2). Then, we show in Lemma A.18 that for many practical applications, this discoverability condition is satisfied.

### A.2.1. Proof of Theorem 4.3

We first define the largest region that **LSE** can safely explore for a given safe initialization $S$ and then we show that we can find the optimum with $\epsilon$-precision within this region. To this end, we define the reachability operator $R^c_\epsilon(S)$ and the fully connected safe region $\bar{R}^c_\epsilon(S)$ by (adapted from [18,21])

$$R^c_\epsilon(S) := S \cup \{a \in \mathcal{A} \mid \exists a' \in S \text{ such that } g_i(a', x_0) - \epsilon - L_a \left\|a - a'\right\| \geq 0, \\ \forall i \in \mathcal{I}_g\}, \tag{A.12}$$

$$\bar{R}^c_\epsilon(S) := \lim_{n\to\infty} \left(R^c_\epsilon\right)^n(S). \tag{A.13}$$

The reachability operator $R^c_\epsilon(S)$ contains the parameters we can safely explore if we know our constraint function with $\epsilon$-precision within some safe set of parameters $S$. Further, $(R^c_\epsilon)^n(S)$ denotes the repeated composition of $R^c_\epsilon(S)$ with itself, and $\bar{R}^c_\epsilon(S)$ its closure. Next, we derive a property for the reachability operator, that we will leverage to provide optimality guarantees.

**Lemma A.11.** *Let $A \subseteq S$, if $\bar{R}^c_\epsilon(A) \setminus S \neq \emptyset$, then $R^c_\epsilon(S) \setminus S \neq \emptyset$.*

**Proof.** This lemma is a straightforward generalization of [18, Lem. 7.4]. Assume $R^c_\epsilon(S) \setminus S = \emptyset$, we want to show that this implies $\bar{R}^c_\epsilon(A) \setminus S = \emptyset$. By definition $R^c_\epsilon(S) \supseteq S$ and therefore $R^c_\epsilon(S) = S$. Iteratively applying $R^c_\epsilon$ to both the sides, we get in the limit $\bar{R}^c_\epsilon(S) = S$. Furthermore, because $A \subseteq S$, we have $\bar{R}^c_\epsilon(A) \subseteq \bar{R}^c_\epsilon(S)$ [18, Lem. 7.1]. Thus, we obtain $\bar{R}^c_\epsilon(A) \subseteq \bar{R}^c_\epsilon(S) = S$, which leads to $\bar{R}^c_\epsilon(A) \setminus S = \emptyset$. □

In the following, we prove that our **LSE** convergence criterion (see Eq. (10)) guarantees that for the safe initialization $S$, we can explore $\bar{R}^c_\epsilon(S)$ during **LSE** in finite time.

**Theorem A.12.** *Consider any $\epsilon > 0$ and $\delta > 0$. Let Assumptions 2.2 and 2.3 hold, $\beta_n$ be defined as in [18], and $S \subseteq \mathcal{A}$ be an initial safe seed of parameters, i.e., $g(a, x_0) \geq 0$ for all $a \in S$. Assume that the information gain $\gamma_n$ grows sublinearly with $n$ for the kernel $k$. Further let $n^*$ be the smallest integer such that (cdf. the convergence criterion of **LSE** in Eq. (10))*

$$\max_{a\in\mathcal{G}_{n^*-1}\cup\mathcal{M}_{n^*-1}} \max_{i\in\mathcal{I}} w_{n^*-1}(a, i) < \epsilon \text{ and } S_{n^*-1} = S_{n^*}. \tag{A.14}$$

*Then we have that $n^*$ is finite and when running **LSE**, the following holds with probability at least $1 - \delta$ for all $n \geq n^*$:*

$$\bar{R}^c_\epsilon(S) \subseteq S_n, \tag{A.15}$$

$$f(\hat{a}_n) \geq \max_{a\in\bar{R}^c_\epsilon(S)} f(a) - \epsilon, \tag{A.16}$$

*with $\hat{a}_n = \arg\max_{a\in S_n} l_n(a, 0)$.*

**Proof.** We first leverage the result from [18, Thm. 4.1] which provides the following *worst-case* bound on $n^*$

$$\frac{n^*}{\beta_{n^*}\gamma_{|\mathcal{I}|n^*}} \geq \frac{C_1\left(\bar{R}^c_0(S)\right) + 1}{\epsilon^2}, \tag{A.17}$$

where $C_1 = 8/\log(1 + \sigma^{-2})$ and $n^*$ is the smallest integer that satisfies Eq. (A.17). Hence, we have that $n^*$ is finite. The sublinear growth of $\gamma_n$ with $n$ is satisfied for many practical kernels, like the ones we consider in this work [31]. Next, we prove Eq. (A.15). For the sake of contradiction, assume $\bar{R}^c_\epsilon(S) \setminus S_{n^*} \neq \emptyset$. This implies, $R^c_\epsilon(S_{n^*}) \setminus S_{n^*} \neq \emptyset$ (Lemma A.11). Therefore, there exists some $a \in \mathcal{A} \setminus S_{n^*}$ such that for some $a' \in S_{n^*} = S_{n^*-1}$ (Eq. (A.14)), we have for all $i \in \mathcal{I}_g$

$$0 \leq g_i(a', x_0) - \epsilon - L_a \left\|a - a'\right\|, \\ \leq u_{n^*-1}(a', i) - L_a \left\|a - a'\right\|. \tag{Lemma A.9}$$

Therefore, $a' \in \mathcal{G}_{n^*-1}$ (see [18] or Appendix D Definition D.1) and accordingly, $w_{n^*-1}(a', i) < \epsilon$. Next, because $w_{n^*-1}(a', i) < \epsilon$, we have for all $i \in \mathcal{I}_g$

$$0 \leq g_i(a', x_0) - \epsilon - L_a \left\|a - a'\right\| \leq l_{n^*-1}(a', i) - L_a \left\|a - a'\right\|. \tag{A.18}$$

This means $a \in S_{n^*}$ (Eq. (8)), which is a contradiction. Thus, we conclude that $\bar{R}^c_\epsilon(S) \subseteq S_{n^*}$ and because $S_{n^*} \subseteq S_n$ for all $n \geq n^*$ (Proposition A.13), we get $\bar{R}^c_\epsilon(S) \subseteq S_n$.

Now we prove Eq. (A.16). Consider any $n \geq n^*$. Note, $w_{n^*-1}(a', i) < \epsilon$, implies $w_n(a', i) < \epsilon$ (see Algorithm 1 line 4 or Algorithm 2 line 13). For simplicity, we denote the solution of $\arg\max_{a \in \bar{R}^c_\epsilon(S)} f(a)$ as $a^*_S$. We have

$$\begin{aligned}
u_n(a^*_S, 0) &\geq f(a^*_S) && \text{(Lemma A.9)}\\
&\geq f(\hat{a}_n) && \text{(by definition of } a^*_S)\\
&\geq l_n(\hat{a}_n, 0) && \text{(Lemma A.9)}\\
&= \max_{a \in S_n} l_n(a, 0). && \text{(by definition of } \hat{a}_n)
\end{aligned}$$

Therefore, $a^*_S$ is a maximizer, i.e., $a^*_S \in \mathcal{M}_n$ (see Appendix D Definition D.2) and has uncertainty less than $\epsilon$, that is, $w_n(a^*_S, i) < \epsilon$. Now, we show that $f(\hat{a}_n) \geq f(a^*_S) - \epsilon$. For the sake of contradiction assume,

$$f(\hat{a}_n) < f(a^*_S) - \epsilon. \tag{A.19}$$

Then we obtain,

$$\begin{aligned}
l_n(a^*_S, 0) &\leq l_n(\hat{a}_n, 0) && \text{(by definition of } \hat{a}_n)\\
&\leq f(\hat{a}_n) && \text{(Lemma A.9)}\\
&< f(a^*_S) - \epsilon && \text{(by Eq. (A.19))}\\
&\leq u_n(a^*_S, 0) - \epsilon && \text{(Lemma A.9)}\\
&\leq u_n(a^*_S, 0) - w_n(a^*, 0) && \text{(because } w_n(a^*_S, 0) \leq \epsilon)\\
&= l_n(a^*_S, 0), && \text{(by definition of } w_n(a^*_S, 0))
\end{aligned}$$

which is a contradiction. Therefore, we have $f(\hat{a}_n) \geq f(a^*_S) - \epsilon$. □

Theorem A.12 states that for a given safe seed $S$, the convergence of **LSE** (Eq. (10)) implies that we have discovered its fully connected safe region $\bar{R}^c_\epsilon(S)$ and recovered the optimum within the region with $\epsilon$-precision.

Based on the previous results, we can show that if the safe global optimum is discoverable for some iteration $n \geq 0$ (see Definition 4.2), then we can find an approximately optimal safe solution. However, to prove optimality, what we also require is that if $a^* \in S_n$ then $a^* \in S_{n+1}$.

**Proposition A.13.** *Let the assumptions from Theorem 4.1 hold. For any $n \geq 0$, the following property is satisfied for $S_n$.*

$$S_n \subseteq S_{n+1}, \tag{A.20}$$

**Proof.** The safe set provably increases during **LSE** [18, Lem. 7.1]. During **GE**, the safe set is only updated if a new safe parameter is found. The proposed update also has the non-decreasing property (see Algorithm 2, line 13). Hence, we can conclude that $S_n \subseteq S_{n+1}$. □

Proposition A.13 shows that if the safe global optimum $a^* \in S_n$, then $a^* \in S_{n+1}$. Next, we prove that if a new safe region $A$ is added to our safe set $S_n$, we will explore its largest reachable safe set $\bar{R}^c_\epsilon(A)$.

**Lemma A.14.** *Consider any integer $n \geq 0$. Let $S_n$ be the safe set of parameters explored after $n$ iterations of* GoSafeOpt *and let $\beta_n$ be defined as in [18]. Consider $A = S_{n+1} \setminus S_n$. If $A \neq \emptyset$, then there exists a finite integer $\bar{n} > n$ such that $\bar{R}^\epsilon_c(A) \cup \bar{R}^\epsilon_c(S_n) \subseteq S_{\bar{n}}$ with probability at least $1 - \delta$.*

**Proof.** First, if $\bar{R}^\epsilon_c(A) \setminus S_{\bar{n}} = \emptyset$ and $\bar{R}^\epsilon_c(S_n) \setminus S_{\bar{n}} = \emptyset$ then $\bar{R}^\epsilon_c(A) \cup \bar{R}^\epsilon_c(S_n) \subseteq S_{\bar{n}}$. We now show that $\bar{R}^\epsilon_c(A) \setminus S_{\bar{n}} = \emptyset$. Assume that $\bar{R}^\epsilon_c(A) \setminus S_{\bar{n}} \neq \emptyset$. We know that $A \subseteq S_{n+1} \subseteq S_{\bar{n}}$ (Proposition A.13). This implies $R^c_\epsilon(S_{\bar{n}}) \setminus S_{\bar{n}} \neq \emptyset$ (Lemma A.11). Since $A \neq \emptyset$, the safe set is expanding. For GoSafeOpt, this can either happen during **LSE** or during **GE** when a new parameter is successfully evaluated, i.e., the boundary condition is not triggered. In either case, we perform **LSE** till convergence. Let $\bar{n} > n$ be the smallest integer for which we converge during **LSE**, i.e., for which

$$\max_{a \in \mathcal{G}_{\bar{n}-1} \cup \mathcal{M}_{\bar{n}-1}} \max_{i \in \mathcal{I}} w_{\bar{n}-1}(a, i) < \epsilon \text{ and } S_{\bar{n}-1} = S_{\bar{n}} \tag{A.21}$$

holds. From Theorem A.12, we know that $\bar{n}$ is finite. Consider $a \in R^c_\epsilon(S_{\bar{n}}) \setminus S_{\bar{n}}$. Then we have that there exists $a' \in S_{\bar{n}}$ such that $0 \leq g_i(a', x_0) - \epsilon - L_a \|a - a'\|$ (see Eq. (A.12)). Furthermore, $S_{\bar{n}-1} = S_{\bar{n}}$, means $a' \in S_{\bar{n}-1}$. Hence, we also have $0 \leq u_{\bar{n}-1}(a, i) - L_a \|a - a'\|$, which implies that, $a' \in \mathcal{G}_{\bar{n}-1}$ (Appendix D Definition D.1) and therefore, $w_{\bar{n}-1}(a', i) < \epsilon$. This implies that $0 \leq l_{\tilde{n}-1}(a', i) - L_a \left\| a - a' \right\|$. Therefore, according to Eq. (8), $a \in S_{\tilde{n}}$, which is a contradiction. Hence, $\bar{R}^{\epsilon}_{c}(A) \setminus S_{\tilde{n}} = \emptyset$. We can proceed similarly to show that $\bar{R}^{\epsilon}_{c}(S_n) \setminus S_{\tilde{n}} = \emptyset$. Since we have $\bar{R}^{\epsilon}_{c}(A) \setminus S_{\tilde{n}} = \emptyset$ and $\bar{R}^{\epsilon}_{c}(S_n) \setminus S_{\tilde{n}} = \emptyset$, we can conclude that $\bar{R}^{\epsilon}_{c}(A) \cup \bar{R}^{\epsilon}_{c}(S_n) \subseteq S_{\tilde{n}}$. □

In Lemma A.14 we have shown that for every set $A$ that we add to our safe set, we will explore its fully connected safe region in finite time. This is crucial because it allows us to guarantee that when we discover a new region during **GE**, we explore it till convergence. Finally, we can now prove Theorem 4.3.

**Theorem 4.3.** *Let $a^*$ be a safe global optimum. Further, let Assumptions 2.1 – 2.5 hold, $\beta_n$ be defined as in [18]. Assume there exists a finite integer $\tilde{n} \geq 0$ such that $a^*$ is discoverable at iteration $\tilde{n}$ (see Definition 4.2). Then, for any $\epsilon > 0$, and $\delta \in (0, 1)$, there exists a finite integer $n^* \geq \tilde{n}$ such that with probability at least $1 - \delta$,*

$$f(\hat{a}_n) \geq f(a^*) - \epsilon, \quad \forall n \geq n^* \tag{13}$$

*with $\hat{a}_n = \arg\max_{a \in S_n} l_n(a, 0)$.*

**Proof.** Since, $a^*$ is discoverable at iteration $\tilde{n}$, there exists a set $A^* \subseteq S_{\tilde{n}}$ such that $a^* \in \bar{R}^{c}_{\epsilon}(A^*)$. Furthermore, we have $\bar{R}^{c}_{\epsilon}(A^*) \subseteq \bar{R}^{c}_{\epsilon}(S_{\tilde{n}})$ (Lemma A.14), therefore, $a^* \in \bar{R}^{c}_{\epsilon}(S_{\tilde{n}})$. Theorem A.12 shows that we can find the optimum in the safe region with $\epsilon$ precision in finite time $n^* \geq \tilde{n}$. Hence, there exists a finite integer $n^*$ such that

$$f(\hat{a}_n) \geq f(a^*) - \epsilon, \quad \forall n \geq n^*, \tag{A.22}$$

with $\hat{a}_n = \arg\max_{a \in S_n} l_n(a, 0)$. □

*A.2.2. Requirements for discovering safe sets with **GE***

In the previous section, we showed that if a safe global optimum $a^*$ is discoverable at some iteration $\tilde{n}$, we can then find it with $\epsilon$-precision. In this section, we show that if for a parameter $a_{\mathbf{GE}}$ in $\mathcal{A} \setminus S_n$, we have backup policies for all the states in its trajectory, then $a_{\mathbf{GE}}$ will be eventually added to our safe set of parameters. Finally, we conclude this section by showing that for many practical cases, $a^*$ fulfills the discoverability condition.

Now, we derive conditions that allow us to explore new regions/parameters during **GE**. To this end, we start by defining a set of safe states $\mathcal{X}^s_n$, i.e., the states for which our boundary condition does not trigger a backup policy.

**Definition A.15.** The set of safe states $\mathcal{X}^s_n$ is defined as

$$\mathcal{X}^s_n := \bigcup_{(a', x') \in \mathcal{B}_n} \left\{ x \in \mathcal{X} \,\middle|\, \left\| x' - x \right\| \leq \frac{1}{L_x} \min_{i \in \mathcal{I}_g} l_n(a', i) - \Xi, \right\}. \tag{A.23}$$

Intuitively, if a trajectory induced by a parameter being evaluated during **GE** lies in $\mathcal{X}^s_n$, then the boundary condition will not be triggered for this parameter. Now we will prove that this set of safe states $\mathcal{X}^s_n$ is non-decreasing. This is an important property because it tells us that GoSafeOpt continues to learn backup policies for more and more states.

**Lemma A.16.** *Let the assumptions from Theorem 4.1 hold. For any $n \geq 0$, the following property is satisfied for $\mathcal{X}^s_n$.*

$$\mathcal{X}^s_n \subseteq \mathcal{X}^s_{n+1}. \tag{A.24}$$

**Proof.** The lower bounds $l_n(a, i)$ are non-decreasing for all $i \in \mathcal{I}$ by definition (see Algorithm 1 line 4 or Algorithm 2 line 13). Additionally, because we continue to add new rollouts to our set of backups, we have $\mathcal{B}_n \subseteq \mathcal{B}_{n+1}$ (see Algorithm 1 line 3 or Algorithm 2 line 12). For each $x \in \mathcal{X}^s_n$, there exists $(a_s, x_s) \in \mathcal{B}_n$, such that $l_n(a_s, i) - L_x (\|x - x_s\| + \Xi) \geq 0$ for all $i \in \mathcal{I}_g$. Because $\mathcal{B}_n \subseteq \mathcal{B}_{n+1}$ and $l_{n+1}(a_s, i) \geq l_n(a_s, i)$, $x \in \mathcal{X}^s_{n+1}$. □

Next, we state conditions under which a parameter $a_{\mathbf{GE}} \in \mathcal{A} \setminus S_n$ will be discovered during **GE**, i.e., no backup policy would be triggered during **GE**, in finite time.

**Lemma A.17.** *Consider any $n \geq 0$. Let $S_n$ be the safe set of parameters explored after $n$ iterations of GoSafeOpt and $a_{\mathbf{GE}}$ a parameter in $\mathcal{A} \setminus S_n$. Further, let the assumptions from Theorem 4.3 hold and $\beta_n$ be defined as in [18]. If, for all $k \geq 0$, $x^{a_{\mathbf{GE}}}(k) \in \mathcal{X}^s_n$, where, $x^{a_{\mathbf{GE}}}(k)$ represents the state at time step $k$ for the system starting at $x_0$ with policy $\pi^{a_{\mathbf{GE}}}(\cdot)$, then there exists a finite integer $\tilde{n} > n$, such that $a_{\mathbf{GE}} \in S_{\tilde{n}}$.*

**Proof.** Assume that there exists no finite integer $\tilde{n} > n$ such that $a_{\mathbf{GE}} \in S_{\tilde{n}}$. This would imply that $a_{\mathbf{GE}} \in \mathcal{A} \setminus S_{\tilde{n}}$ for all $\tilde{n} > n$. Thus, because $a_{\mathbf{GE}}$ will never be a part of our safe set, it will never be evaluated during **LSE**. However, from Theorem A.12 we know that **LSE** will converge in a finite number of iterations after which we will perform **GE**. Since $a_{\mathbf{GE}}$ is not a part of the safe set, it can only be evaluated during **GE**, where parameters outside of the safe regions are queried. The parameter space $\mathcal{A}$ is finite and any parameter that was evaluated unsuccessfully, i.e., boundary condition was triggered, will be added to $\mathcal{E}$ and therefore not evaluated again (see Eq. (11) and Algorithm 2 line 9). This implies that $a_{\mathbf{GE}}$ will be evaluated for some $n'$ with $n < n' < \tilde{n}$ (since $S_{n'} \subseteq S_{\tilde{n}}$, see Proposition A.13). Furthermore, $\mathcal{X}^s_n \subseteq \mathcal{X}^s_{n'}$ (Lemma A.16) and, therefore, for all $k \geq 0$, $x^{a_{\mathbf{GE}}}(k) \in \mathcal{X}^s_{n'}$. If when $a_{\mathbf{GE}}$ is evaluated, the experiment is unsuccessful, i.e., we were to trigger a backup policy, this would imply that for some $k'$ and $i \in \mathcal{I}_g$, there is no $(a_s, x_s) \in \mathcal{B}_{n'}$ such that

$$l_{n'}(a, i) \geq L_x(\|x^{a_{\mathbf{GE}}}(k') - x_s\| + \Xi).$$

Thus, we had $x^{a_{\mathbf{GE}}}(k') \notin \mathcal{X}^s_{n'}$, which contradicts our assumption. Therefore, $a_{\mathbf{GE}} \in S_{n'+1} \subseteq S_{\tilde{n}}$ (Proposition A.13), which is a contradiction. Consequently, we have $a_{\mathbf{GE}} \in S_{\tilde{n}}$ after a finite number of iterations $\tilde{n}$. $\square$

Lemma A.17 guarantees that, if the trajectory of parameter $a_{\mathbf{GE}}$ lies in $\mathcal{X}^s_n$, then it will eventually be added to our safe set, either during **LSE** or during **GE**. We utilize this result to provide a condition for the safe global optimum $a^*$ to be *discoverable* at iteration $\tilde{n}$ by GoSafeOpt.

**Lemma A.18.** *Consider any $n \geq 0$. Let Assumptions 2.1 – 2.4 hold, $\beta_n$ be defined as in [18], and let $a^* \in \mathcal{A}$ be the safe global optimum. If there exists $a_{\mathbf{GE}} \in \mathcal{A} \setminus S_n$ such that*

*(i) for all $k \geq 0$ and some $n \geq 0$, $x^{a_{\mathbf{GE}}}(k) \in \mathcal{X}^s_n$, where $x^{a_{\mathbf{GE}}}(k)$ is the state visited by the system starting at $x_0$ under the policy $\pi^{a_{\mathbf{GE}}}(\cdot)$ at time step k, and*
*(ii) $a^* \in \bar{R}^c_\epsilon(\{a_{\mathbf{GE}}\})$,*

*then $a^*$ is discoverable at iteration $\tilde{n} \geq n$ by GoSafeOpt with probability at least $1 - \delta$.*

**Proof.** Since, for all $k \geq 0$, $x^{a_{\mathbf{GE}}}(k) \in \mathcal{X}^s_n$, there exists a finite integer $\tilde{n} > n$ such that $a_{\mathbf{GE}} \in S_{\tilde{n}}$ (Lemma A.17). Furthermore, because $a^* \in \bar{R}^c_\epsilon(\{a_{\mathbf{GE}}\})$, we can conclude that $a^*$ is discoverable at iteration $\tilde{n}$ (see Definition 4.2). $\square$

The condition here is interesting because empirically, for many practical cases, it is fulfilled. Crucially, optimal or near-optimal parameters tend to visit similar states as other safe policies. We add rollouts from safe policies to our set of backups $\mathcal{B}_n$ and therefore have backup policies for their trajectories and other trajectories that lie close to them. Therefore, the trajectories of (near-)optimal parameters lie in $\mathcal{X}^s_n$ and in this case, the safe global optimum fulfills the discoverability condition from Theorem 4.3.

## Appendix B. Additional information on experiments

For all our experiments in Section 5, we consider a controller in the operational space. The operational space dynamics of the end-effector are given by [36]

$$u(x(t)) = \Lambda(q)\ddot{s} + \Gamma(q, \dot{q})\dot{s} + \eta(q), \tag{B.1}$$

where $s$ represents the end-effector position, $q$ the joint angles, and $\Lambda(q)$, $\Gamma(q, \dot{q})$, $\eta(q)$ are nonlinearities representing the mass, Coriolis, and gravity terms, respectively. The state we consider is $x(t) = [s^T(t), \dot{s}^T(t)]^T$. We apply an impedance controller:

$$u\,(x(t)) = -K(x - x_{\text{des}}(k)) + \Gamma(q, \dot{q})\dot{s} + \eta(q), \tag{B.2}$$

with $K$ being the feedback gain. The torque $\tau$ applied to each of the joints can be calculated via $\tau = J^T u(x(t))$, with $J$ the Jacobian.

For our experiments, we can directly measure $g(a, x(k))$, where $k$ denotes a discrete time step. Therefore, instead of using $l_n(a_s, i)$ for the boundary condition in Section 4.3.2, we take a lower bound over all the tuples in our set of backups, $\mathcal{B}_n$, i.e., $l_n(a_s, x_s, i)$, which could potentially reduce the conservatism of the boundary condition. Therefore, we define a GP over the parameter and state space, which contains all the points from $\mathcal{B}_n$. The set $\mathcal{B}_n$ consists of rollouts from individual experiments, we typically add $50 - 100$ data points from each experiment to $\mathcal{B}_n$. As the data points of our GP increase, inference becomes prohibitively costly. To this end, we use a subset selection scheme to select a small subset of points $(a, x)$ from $\mathcal{B}_n$ at random with a probability that is proportional to $\exp(-\min_{i\in\mathcal{I}_g} l^2_n(a, x, i))$. Crucially, we want to retain points that have a small lower bound such that we have low uncertainty around these points. We perform this subset selection once our GP has acquired more than $n_{\max}$ data points. Then, we select a subset of $m < n_{\max}$ points. Lastly, as described in Section 5.2.1, for the boundary condition from Section 4.3.2, we define the distances $d_u$, $d_l$ using covariances $\kappa_u$, $\kappa_l$, respectively. Particularly, we pick $d_u$ such that $k(d_u) \geq \kappa_u$ for the stationary isotropic kernel $k$ that we use to model our GP (same for $d_l$). This makes the choice of $d_u$ more intuitive since it directly relates to the covariance function of our GP.

## B.1. Simulation

For the simulation task, we determine the impedance $K$ using an infinite horizon LQR parameterized via

$$Q = \begin{bmatrix} Q_r & 0 \\ 0 & \kappa_d Q_r \end{bmatrix}, Q_r = 10^{q_c} I_3,$$
$$R = 10^{r-2} I_3, A = \begin{bmatrix} 0 & I_3 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ I_3 \end{bmatrix}.$$

The matrices $A$, $B$ are obtained assuming that we use a feedback linearization controller [36]. However, because we instead use an impedance controller, there are nonlinearities and imprecisions in our model. The parameters $q_c, r, \kappa_d$ are tuning parameters we would like to optimize. We define the desired path $x_{\text{des}}(k)$ as

$$x_{\text{des}}(k) = [p_0^T + \frac{k}{T_{traj}}(p_{\text{des}} - s_0)^T, 0_{1x3}]^T \quad \text{(8D task)}$$
$$x_{\text{des}}(k) = x_{\text{des}}(\rho(k)) \quad \text{(11D task)}$$

Here, $\rho(\cdot)$ is used to parameterize a cubic spline from $x_0$ to $x_{\text{target}}$. The constraint is:

$$\bar{g}(x(t)) = \frac{\|s(t) - s_{\text{des}}\|_2 - \|s(0) - s_{\text{des}}\|_2}{\|s(0) - s_{\text{des}}\|_2} - \alpha, \quad \alpha = 0.08 \quad \text{(8D task)}$$
$$\bar{g}\big(x(t)\big) = \zeta - \big\|x(t) - x_{\text{d}}\big(\rho(t)\big)\big\|_2. \quad \text{(11D task)}$$

The stage rewards, i.e., rewards received at each time step [1], are

$$\mathcal{R}(x(t)) = -\|s(t) - s_{\text{des}}\|_2^2 / \|s(0) - s_{\text{des}}\|_2^2 - \frac{1}{25}\|\tanh \dot{s}(t)\|_2^2 - \frac{1}{25}\|\tanh u(x(t))\|_2^2 \quad \text{(8D task)}$$
$$\mathcal{R}\big(x(t)\big) = -\nu_\rho(\rho(t) - 1)^2 - \nu_x \big\|x(t) - x_{\text{d}}\big(\rho(t)\big)\big\|_2 - \nu_u \left\|\frac{u}{u_{\max}}\right\|_2. \quad \text{(11D task)}$$

Additionally, to encourage fast behavior in the eight-dimensional task, we only sample parameters for which the eigenvalues of $A - BK$ are less than a fixed threshold: $\text{eig}(A - BK) \leq -10$. Although this constraint is independent of the state, it can be evaluated before each experiment and parameters can be rejected if the criterion is not fulfilled. The value for $\kappa_d$ is heuristically set to 0.1 for the first experiment (8D task). For the eleven-dimensional task, $\kappa_d$ is also tuned. In the eight-dimensional task, we observe that the underlying functions $f, g_i$ exhibit non-smooth behavior. Therefore we use the Matérn kernel [30] with parameter $\nu = 3/2$ for our GP. For the remaining tasks, we use the squared exponential (SE) kernel.

## B.2. Hardware

For the hardware task, we define the subsequent objective and constraint functions:

$$\mathcal{R}(x(t)) = -\|s(t) - s_{\text{des}}(t)\|_2,$$
$$\bar{g}(s(t)) = \|s(t) - s_w\|_{P,\infty} - \psi,$$

where $s_w$ represents the center of the wall in Fig. 4 and $\|s - s_w\|_{P,\infty} \leq d_w$ defines the rectangular shaped outline around the wall and $\psi > d_w$.

## Appendix C. Sensitivity to LSE and GE steps

We analyze the sensitivity of the practical version of GoSafeOpt with respect to $n_{\textbf{LSE}}$ and $n_{\textbf{GE}}$ on a simple one-dimensional toy example. The toy example consists of a one-dimensional system that has the following dynamics, stage reward, and constraint:

$$s(k+1) = 1.01\sqrt{|s(k)|} - 0.2\sqrt{|ay(k)|} + v(k), \quad \text{with } y(k) = s(k) + w(k)$$
$$\mathcal{R}(s) = -s^2,$$
$$\bar{g}(s) = s^2 - 0.81,$$

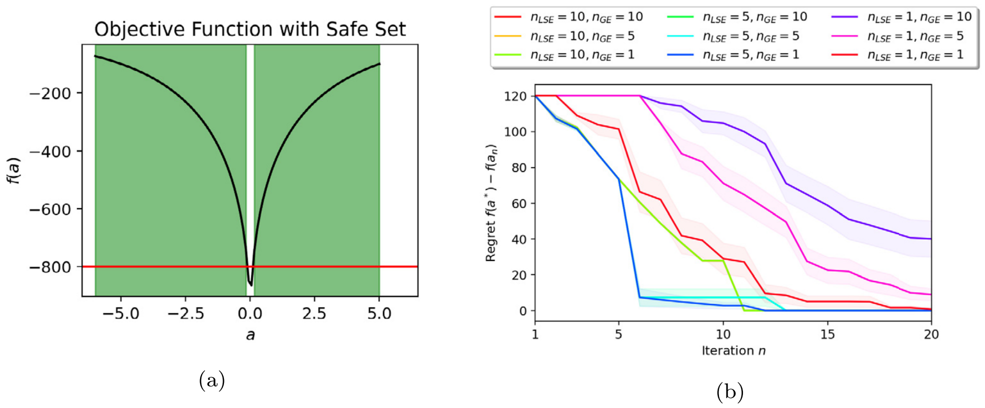

**Fig. C.11.** Toy Example results. *(a) Objective function with the safe set. (b) Performance of* GoSafeOpt *for different* $n_{\mathbf{LSE}}$ *and* $n_{\mathbf{GE}}$.

with $v(k), w(k) \sim \mathcal{N}(0, 10^{-4})$ and $a \in [-6, 5]$ the control parameter we would like to optimize. We consider a regulation problem, i.e. we start at $x_0 = 0$ and we would like our system to remain close to $x_0$. We run GoSafeOpt for twenty iterations over twenty seeds for different $n_{\mathbf{LSE}}$ and $n_{\mathbf{GE}}$ values. Fig. C.11 depicts the safe set and the performance of GoSafeOpt for different $n_{\mathbf{LSE}}$ and $n_{\mathbf{GE}}$. In this example, we have two disconnected safe sets and $a^* = -6$ is the global optimum. To show the advantage of global exploration, we initialize GoSafeOpt in the safe region which does not contain $a^*$. For all our experiments, we obtain 100% safety. Furthermore our results show, for $n_{\mathbf{LSE}} = 5$, we get faster convergence than for $n_{\mathbf{LSE}} = 10$. This is because we explore the region with the global optimum earlier. However, when we do not perform enough **LSE** steps then global exploration also fails and we are stuck at a bad optimum (see for instance $n_{\mathbf{LSE}} = 1$ and $n_{\mathbf{GE}} = 10$). This simple example highlights the trade-off between local exploration and global exploration steps.

## Appendix D. Additional definitions

In this section, we present some of the definitions from SafeOpt for completeness.

**Definition D.1.** The expanders $\mathcal{G}_n$ are defined as $\mathcal{G}_n := \{a \in S_n \mid e_n(a) > 0\}$ with $e_n(a) = |\{a' \in \mathcal{A} \setminus S_n, \exists i \in \mathcal{I}_g : u_n(a, i) - L_a \left\|a - a'\right\| \geq 0\}|$.

**Definition D.2.** The maximizers $\mathcal{M}_n$ are defined as $\mathcal{M}_n := \{a \in S_n \mid u_n(a, 0) \geq \max_{a' \in S_n} l_n(a', 0)\}$.

## Appendix E. Hyperparameters

The hyperparameters of our simulated and real-world experiments are provided in Table E.2.

**Table E.2**
Table of hyperparameters.

| | 8D task Simulation | | 11D task Simulation | | Hardware task | |
|---|---|---|---|---|---|---|
| | SafeOpt | GoSafeOpt | SafeOptSwarm | GoSafeOpt | SafeOptSwarm | GoSafeOpt |
| Iterations | 200 | 200 | 200 | 200 | 50 | 50 |
| $\beta_n^{1/2}$ | 4 | 4 | 3 | 3 | 3 | 3 |
| $a$ lengthscale | 0.12,0.12 | 0.12,0.12 | 0.067,0.2,0.13,0.2 | 0.067,0.2,0.13,0.2 | 0.1,0.1, 0.1 | 0.1,0.1, 0.1 |
| $\kappa$ for $f$ and $g$ | 1, 1 | 1, 1 | 1, 1 | 1, 1 | 1, 1 | 1, 1 |
| $\sigma$ for $f$ and $g$ | 0.1,0.1 | 0.1,0.1 | 0.1,0.1 | 0.1,0.1 | 0.05,0.3 | 0.05,0.3 |
| $x$ lengthscale | - | 0.3,0.3,0.3, 2.5,2.5,2.5 | - | 0.5,0.5,0.5, 0.6,0.6,0.6, 10 | - | 0.3,0.3,0.3, 0.5,0.5,0.2 |
| $\epsilon$ | - | 0.1 | - | 0.1 | - | 0.01 |
| max **LSE** steps | - | 30 | - | 100 | - | 20 |
| max **GE** steps | - | 10 | - | 10 | - | 5 |
| $\kappa_l$ | - | 0.90 | - | 0.90 | - | 0.90 |
| $\kappa_u$ | - | 0.94 | - | 0.94 | - | 0.94 |
| $\eta_l$ | - | 0.4 | - | 0.3 | - | 0.9 |
| $\eta_u$ | - | 0.6 | - | 0.75 | - | 1.1 |
| $n_{\max}$ | - | 1000 | - | 1000 | - | 1000 |
| $m$ | - | 500 | - | 500 | - | 500 |

## References

[1] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 2nd edition, The MIT Press, 2018.
[2] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (1) (2016) 1334–1373.
[3] J. Peters, S. Schaal, Reinforcement learning of motor skills with policy gradients, Neural Netw. 21 (4) (2008) 682–697.
[4] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint, arXiv:1509.02971, 2015.
[5] J. Kober, J.A. Bagnell, J. Peters, Reinforcement learning in robotics: a survey, Int. J. Robot. Res. 32 (11) (2013) 1238–1274.
[6] S. Schaal, C.G. Atkeson, Learning control in robotics, IEEE Robot. Autom. Mag. 17 (2) (2010) 20–29.
[7] J. Mockus, V. Tiesis, A. Zilinskas, The application of Bayesian methods for seeking the extremum, Towards Global Optim. 2 (117–129) (1978) 2.
[8] R. Calandra, A. Seyfarth, J. Peters, M. Deisenroth, Bayesian optimization for learning gaits under uncertainty, Ann. Math. Artif. Intell. 76 (2016) 5–23.
[9] A. Marco, P. Hennig, J. Bohg, S. Schaal, S. Trimpe, Automatic LQR tuning based on Gaussian process global optimization, in: IEEE International Conference on Robotics and Automation, 2016, pp. 270–277.
[10] R. Antonova, A. Rai, C.G. Atkeson, Deep kernels for optimizing locomotion controllers, in: Conference on Robot Learning, 2017, pp. 47–56.
[11] M. Turchetta, A. Krause, S. Trimpe, Robust model-free reinforcement learning with multi-objective Bayesian optimization, in: IEEE International Conference on Robotics and Automation, 2020, pp. 10702–10708.
[12] M. Gelbart, J. Snoek, R. Adams, Bayesian optimization with unknown constraints, in: Conference on Uncertainty in Artificial Intelligence, 2014, pp. 250–259.
[13] J.M. Hernández-Lobato, M.A. Gelbart, R.P. Adams, M.W. Hoffman, Z. Ghahramani, A general framework for constrained Bayesian optimization using information-based search, J. Mach. Learn. Res. 17 (1) (2016) 5549–5601.
[14] A. Marco, D. Baumann, M. Khadiv, P. Hennig, L. Righetti, S. Trimpe, Robot learning with crash constraints, IEEE Robot. Autom. Lett. 6 (2) (2021) 1439–1446.
[15] S. Heim, A. Rohr, S. Trimpe, A. Badri-Spröwitz, A learnable safety measure, in: Conference on Robot Learning, 2020, pp. 627–639.
[16] Y. Sui, A. Gotovos, J. Burdick, A. Krause, Safe exploration for optimization with Gaussian processes, in: International Conference on Machine Learning, 2015, pp. 997–1005.
[17] F. Berkenkamp, A.P. Schoellig, A. Krause, Safe controller optimization for quadrotors with Gaussian processes, in: IEEE International Conference on Robotics and Automation, 2016, pp. 491–496.
[18] F. Berkenkamp, A. Krause, A.P. Schoellig, Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics, Mach. Learn. (2021).
[19] C. König, M. Turchetta, J. Lygeros, A. Rupenyan, A. Krause, Safe and efficient model-free adaptive control via Bayesian optimization, arXiv preprint, arXiv:2101.07825, 2021.
[20] E.N. Gryazina, B.T. Polyak, Stability regions in the parameter space: D-decomposition revisited, Automatica 42 (1) (2006) 13–26.
[21] D. Baumann, A. Marco, M. Turchetta, S. Trimpe, GoSafe: globally optimal safe robot learning, in: IEEE International Conference on Robotics and Automation, 2021, pp. 4452–4458. Proofs in extended online version: arXiv:2105.13281.
[22] J. Kirschner, M. Mutny, N. Hiller, R. Ischebeck, A. Krause, Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces, in: International Conference on Machine Learning, 2019, pp. 3429–3438.
[23] Y. Sui, V. Zhuang, J. Burdick, Y. Yue, Stagewise safe Bayesian optimization with Gaussian processes, in: International Conference on Machine Learning, 2018, pp. 4781–4789.
[24] K.P. Wabersich, M.N. Zeilinger, A predictive safety filter for learning-based control of constrained nonlinear dynamical systems, Automatica 129 (2021) 109597.
[25] P. Wieland, F. Allgöwer, Constructive safety using control barrier functions, IFAC Proc. Vol. 40 (12) (2007) 462–467.
[26] R. Cheng, G. Orosz, R.M. Murray, J.W. Burdick, End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, in: AAAI Conference on Artificial Intelligence, 2019, pp. 3387–3395.
[27] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.
[28] C. Fiedler, C.W. Scherer, S. Trimpe, Practical and rigorous uncertainty bounds for Gaussian process regression, AAAI Conf. Artif. Intell. 35 (8) (2021) 7439–7447.
[29] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, 2014.
[30] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, Adaptive Computation and Machine Learning, The MIT Press, 2005.
[31] N. Srinivas, A. Krause, S.M. Kakade, M.W. Seeger, Information-theoretic regret bounds for Gaussian process optimization in the bandit setting, IEEE Trans. Inf. Theory 58 (5) (2012) 3250–3265.
[32] S.R. Chowdhury, A. Gopalan, On kernelized multi-armed bandits, in: International Conference on Machine Learning, 2017, pp. 844–853.
[33] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley Series in Telecommunications and Signal Processing, Wiley-Interscience, USA, 2006.
[34] A. Krause, C. Ong, Contextual Gaussian process bandit optimization, in: Advances in Neural Information Processing Systems, 2011.
[35] R.R. Duivenvoorden, F. Berkenkamp, N. Carion, A. Krause, A.P. Schoellig, Constrained Bayesian optimization with particle swarms for safe adaptive controller tuning, in: 20th IFAC World Congress, IFAC-PapersOnLine 50 (1) (2017) 11800–11807.
[36] B. Siciliano, O. Khatib, Springer Handbook of Robotics, 2nd edition, Springer Publishing Company, Incorporated, 2016.
[37] E. Todorov, T. Erez, Y. Tassa, Mujoco: a physics engine for model-based control, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5026–5033.
[38] D.P. Bertsekas, Dynamic Programming and Optimal Control, 2nd edition, Athena Scientific, 2000.
[39] A. Wischnewski, J. Betz, B. Lohmann, A model-free algorithm to safely approach the handling limit of an autonomous racecar, in: IEEE International Conference on Connected Vehicles and Expo, 2019, pp. 1–6.
[40] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, A. Krause, Safe contextual Bayesian optimization for sustainable room temperature PID control tuning, in: International Joint Conference on Artificial Intelligence, 2019, pp. 5850–5856.
[41] C. König, M. Turchetta, J. Lygeros, A. Rupenyan, A. Krause, Safe and efficient model-free adaptive control via Bayesian optimization, in: IEEE International Conference on Robotics and Automation, 2021, pp. 9782–9788.
[42] S.E. Cooper, T.I. Netoff, Multidimensional Bayesian estimation for deep brain stimulation using the safeopt algorithm, medRxiv (2022).
[43] J. Rothfuss, C. Koenig, A. Rupenyan, A. Krause, Meta-learning priors for safe Bayesian optimization, arXiv preprint, arXiv:2210.00762, 2022.
[44] F. Berkenkamp, A.P. Schoellig, A. Krause, No-regret Bayesian optimization with unknown hyperparameters, J. Mach. Learn. Res. (2019) 1–24.
[45] A. Schperberg, S.D. Cairano, M. Menner, Auto-tuning of controller and online trajectory planner for legged robots, IEEE Robot. Autom. Lett. (2022).