Efficient sample selection for safe learning

Accepted for IFAC World Congress 2023, Yokohama, JAPAN

Efficient sample selection for safe learning ⋆

Marta Zagorowska * **Efe C. Balta** * **Varsha Behrunani** *,** **Alisa Rupenyan** * **John Lygeros** *

* *Automatic Control Laboratory, ETH Zurich, Switzerland, (E-mails: {mzagorowska,ebalta,bvarsha,ralisa,lygeros}@control.ee.ethz.ch)*
** *Urban Energy Systems Laboratory, Swiss Federal Laboratories for Materials Science and Technology (EMPA), Dübendorf, Switzerland*

Abstract: Ensuring safety in optimization is challenging if the underlying functional forms of either the constraints or the objective function are unknown. The challenge can be addressed by using Gaussian processes to provide confidence intervals used to find solutions that can be considered safe. To iteratively find a trade-off between finding the solution and ensuring safety, the SafeOpt algorithm builds on algorithms using only the upper bounds (UCB-type algorithms) by performing an exhaustive search on the entire search space to find a safe iterate. That approach can quickly become computationally expensive. We reformulate the exhaustive search as a series of optimization problems to find the next recommended points. We show that the proposed reformulation allows using a wide range of available optimization solvers, such as derivative-free methods. We show that by exploiting the properties of the solver, we enable the introduction of new stopping criteria into safe learning methods and increase flexibility in trading off solver accuracy and computational time. The results from a non-convex optimization problem and an application for controller tuning confirm the flexibility and the performance of the proposed reformulation.

Keywords: Machine learning in modelling, prediction, control and automation; Learning for control; Bayesian methods; Safe learning; Derivative-free optimization

1. INTRODUCTION

Ensuring safety in industrial control systems is usually done by enforcing constraints at the design stage of the control algorithm. However, ensuring safety in an online fashion is challenging because the constraints may be unknown. Safe learning algorithms, such as SafeOpt from Sui et al. (2015), use surrogate models to find a safe controller. However, safe learning algorithms based on surrogate models can become computationally expensive. In this paper, we extend the SafeOpt algorithm from Sui et al. (2015) to improve the performance of the algorithm and increase flexibility.

The SafeOpt algorithm proposed by Sui et al. (2015) is an iterative algorithm that uses Gaussian processes to learn unknown functional form of the objective and the constraints. It builds on the algorithm proposed by Srinivas et al. (2012) where the upper bound of the Gaussian process was considered to find the iterates (UCB-GP), without explicitly considering their safety. SafeOpt ensures that in every iteration the new safe points that potentially maximize the objective are chosen based on confidence intervals from the Gaussian processes. The choice of the new points is done by analysing the safety of a selected number of points from the whole search space. This approach is appealing because it does not rely on derivative information, which may often be unavailable. However, looking at the whole search space is equivalent to performing an exhaustive search and can quickly become computationally expensive, as indicated by Berkenkamp et al. (2021). We reformulate the exhaustive search as a series of optimization problems to find the next recommended points.

A review of safe learning methods related to SafeOpt was done by Kim et al. (2021). Most of these algorithms also use discretized search space to find the optimum. However, already Fiducioso et al. (2019) identified the issue of computational effort related to looking at the whole search space in SafeOpt. The idea of merging derivative-free optimization with safe learning was explored by Duivenvoorden et al. (2017) who proposed to find the recommended point by solving auxiliary optimization problems with particle swarm methods. They preserved the idea of SafeOpt to use confidence intervals of Gaussian processes in every iteration, but redefined the way of choosing new points to make it suitable for particle swarm methods. Due to the heuristic nature of particle swarm methods, their approach needed auxiliary adjustments to ensure good performance of the swarm. The reformulation we are proposing in the current paper allows avoiding heuristics while preserving the way the new points are chosen in every iteration of SafeOpt, at the same time improving the computational performance of the algorithm.

To preserve the character of SafeOpt, we solve the reformulated optimization problems using direct search meth-

⋆ Research supported by NCCR Automation, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (grant no. 180545), and by the European Research Council (ERC) under the H2020 Advanced Grant no. 787845 (OCAL).

ods. Similarly to particle swarm methods, direct-search methods belong to derivative-free methods. In contrast to particle swarm methods, direct search provides explicit stopping criteria with convergence guarantees (Audet and Hare, 2017, Ch. 2.). In the current paper, we exploit the properties of pattern search to emulate the discretized structure of the search space required by SafeOpt. We demonstrate with numerical examples that the proposed reformulation mitigates the computational effort while preserving safety.

The rest of the paper is structured as follows. Section 2 presents the necessary background and introduces the SafeOpt algorithm. The reformulation of the algorithm as a series of optimization problems is shown in Section 3. Numerical examples in Section 4 show the performance of the reformulated algorithm. The paper ends with discussion in Section 5.

2. BACKGROUND

SafeOpt from Sui et al. (2015) is designed to solve constrained optimization problems of the form:

$$\max_{x} \quad f(x) \tag{1a}$$

$$\text{subject to} \quad g_j(x) \geq 0,\ j = 1, \ldots, J. \tag{1b}$$

where $x \in \mathcal{A} \subset \mathbb{R}^n$ is a vector of decision variables, $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be maximized, $g_j : \mathbb{R}^n \to \mathbb{R}$ is one of J constraints that must be satisfied. It is assumed that the functional form of neither f nor g_j is known, but there exist oracles that can provide noise-corrupted values of f and g_j. As a result, they can be treated as outputs, whereas x can be treated as an input in Gaussian process regression.

2.1 Gaussian processes

Following Berkenkamp et al. (2016), we use Gaussian processes to approximate both the objective and the constraints using measurements, i.e. we find approximations $J_i(x) : \mathcal{A} \to \mathbb{R}$ where $i = 0$ corresponds to the objective function, $i = 1, \ldots, J$ corresponds to the constraints (1b). Gaussian process regression assumes that the values $J(x_0), J(x_1), \ldots, J(x_P)$ corresponding to different x are random variables, with joint Gaussian distribution for any finite P. The prior information about the functions J_i is defined by the mean and the covariance function $k(x_i, x_j)$.

Assume that we have access to noisy measurements $\hat{J}_i(x) = J(x) + \omega$ where $\omega \sim \mathcal{N}(0, \sigma_\omega^2)$. To use Gaussian processes corresponding to J_i, we need to predict the value of J_i at an arbitrary point $\hat{x}$ using only R past measurement data $\mathbf{J}_i = [\hat{J}_i(x_r)]_{r=1,\ldots,R}$. From Berkenkamp et al. (2016), the mean and the variance of the prediction are:

$$\mu_i(\hat{x}) = \mathbf{k}_R(\hat{x})(\mathbf{K}_R + \mathbf{I}_R\sigma_\omega^2)^{-1}\mathbf{J}_i \tag{2}$$

$$\sigma_{R,i}^2(\hat{x}) = k(\hat{x}, \hat{x}) - \mathbf{k}_R(\hat{x})(\mathbf{K}_R + \mathbf{I}_R\sigma_\omega^2)^{-1}\mathbf{k}_R^\intercal(\hat{x}) \tag{3}$$

where $\mathbf{J}_i$ is a vector of R observed noisy values, $i = 0, \ldots, J$, the matrix $\mathbf{K}_R$ contains the covariances of past data, $k(x_a, x_b)$, $a, b = 1, \ldots, R$, and $\mathbf{k}_R(\hat{x})$ contains the covariances between the new point and the past data. The identity matrix of size $R \times R$ is denoted by $\mathbf{I}_R$.

The mean and the variance are then used to find the lower and upper confidence bounds:

$$\begin{aligned} l_R(x, i) =& \mu_i(x) - \beta\sigma_{R,i}(x), \\ u_R(x, i) =& \mu_i(x) + \beta\sigma_{R,i}(x) \end{aligned} \tag{4}$$

where β corresponds to the desired confidence level.

2.2 SafeOpt

Default SafeOpt The idea of using confidence bounds for optimization in unconstrained optimization was analyzed by Srinivas et al. (2012). The current paper is based on the version of SafeOpt proposed by Berkenkamp et al. (2016), which includes five steps. The algorithm requires an *initial safe set* set S_0 containing a number of initial points, such that the constraints (1b) are satisfied (Step 1). Using S_0, from (4), we obtain the upper and lower bound corresponding to the initial safe set (Step 2). The safe set in iteration n is defined as:

$$S_n = \bigcap_{i=1,\ldots,J} \{x \in A : l_n(x, i) \geq J_{\min}\} \tag{5}$$

where the parameter $J_{\min} \geq 0$ is a design parameter that defines safety (Berkenkamp et al., 2016) and $A \subset \mathcal{A}$ is a discrete search space. The algorithm then looks for a recommended point $x_n \in S_n$ by finding a trade-off between *maximizers* corresponding to the potential optimum and *expanders* related to extending the current safe set S_n (Step 3).

The set of maximizers M_n is defined as:

$$M_n = \{x \in S_n : u_n(x, 0) \geq \max_{x \in S_n} l_n(x, 0)\}. \tag{6}$$

Berkenkamp et al. (2016) define the expanders G_n as:

$$G_n = \{x \in S_n : |\mathcal{G}(x)| > 0\} \tag{7}$$

where $|\cdot|$ represents the cardinality of a set and

$$\mathcal{G}(\overline{x}) = \{x' \in A \setminus S_n : \forall j\ l_{n,j,(\overline{x},u_n(\overline{x},j))}(x') \geq J_{\min}\}. \tag{8}$$

where $l_{n,(\overline{x},u_n(\overline{x}))}(x)$ the lower bound for the point x obtained from the auxiliary GP calculated for a given point $\overline{x} \in S_n$. The auxiliary GP is created assuming that we observed the optimistic upper bound $u_n(\overline{x}, i)$ (Berkenkamp et al., 2016).

The new point is chosen as:

$$x_n = \operatorname*{argmax}_{x \in G_n \cup M_n} \max_i w_n(x, i) \tag{9}$$

where $w_n(x, i) = u_n(x, i) - l_n(x, i)$ (Step 4). The point x_n is then applied to the system to collect new observations about the constraints and the objective, and update the Gaussian processes using (2) and (3) (Step 5).

The optimum after n iterations is found as:

$$x^* = \arg\max_{x \in S_n} l_n(x, 0). \tag{10}$$

Analysis of convergence and safety guarantees of the algorithm was done by Berkenkamp et al. (2016).

Discretization of the search space Equation (9) describes searching for a recommended point x_n in the search space defined by the union of the set of expanders G_n and maximizers M_n. The search space $G_n \cup M_n$ is always bounded if the overall search space $\mathcal{A}$ is bounded. If the search space $G_n \cup M_n$ is also discrete, the value of x_n from (9) can be found by exhaustive search in the set

$G_n \cup M_n$. The number of discretization points N will then define how big the search space for the exhaustive search is. Because of the need for an exhaustive search, finding x_n becomes a challenge for large values of N because it imposes a significant computational burden (Berkenkamp et al., 2021). To avoid the choice of discretization and get more flexibility in setting the accuracy, we reformulate the SafeOpt algorithm for finding the recommended value from (9) as a series of optimization problems while preserving the definitions of the sets of maximisers (6) and the expanders (7).

2.3 Pattern search algorithm

Pattern search methods belong to the group of direct search optimization methods and rely on evaluating a number of candidate points around a selected point, which are chosen following a given *pattern*. Given a selected point $x^k \in \mathbb{R}^n$, a *pattern* defines a set of vectors in $\mathbb{R}^n$ where the algorithm looks for an incumbent point. Formally, the algorithm uses a *mesh*, defined as (Audet and Hare, 2017):

$$\mathcal{M}^k := \{x^k + \delta^k D y : y \in \mathbb{N}^p\} \tag{11}$$

where $\delta^k > 0$ describes the mesh size of $\mathcal{M}^k$, and $D = GZ$ where $G \in \mathbb{R}^{n\times n}$ is an invertible matrix, $Z \in \mathbb{R}^{n\times p}$ is such that the columns of Z form a positive spanning set in $\mathbb{R}^n$. Let us denote with $\mathbb{D}$ the columns of D. A positive spanning set $\mathbb{D}^k \subseteq \mathbb{D}$ is called a *pattern*. An in-depth description of the algorithm was provided by Audet and Hare (2017).

Pattern search evaluates the points defined by the current mesh to find x^{k+1} that improves the value of the objective function while satisfying constraints (Lewis and Torczon, 2002). The mesh defined by (11) can be intuitively understood as local discretization around the current point x^k. The algorithm stops if the mesh size becomes smaller than a given threshold $\delta^k \leq \varepsilon$. We explore the flexibility provided by using the mesh size as a stopping criterion in pattern search to introduce new stopping criteria for the reformulated SafeOpt algorithm in Section 4. An illustration is provided for two-dimensional search space in Fig. 1a. The mesh $\mathcal{M}_\mathcal{B}$ (dashed) has been obtained by decreasing the size δ^k of mesh $\mathcal{M}_A$ (solid) by 0.5.

3. SAFEOPT AS OPTIMIZATION

We now present the main result of this paper. We solve two separate optimization problems to find the new recommended value x_n:

$$P_1: \quad \max_{x\in M_n} \max_i w_n(x,i), \tag{12}$$

$$P_2: \quad \max_{x\in G_n} \max_i w_n(x,i) \tag{13}$$

The new recommended value x_n is obtained as the point from $G_n \cup M_n$ such that $w(x_n) = w_{\max}$ with

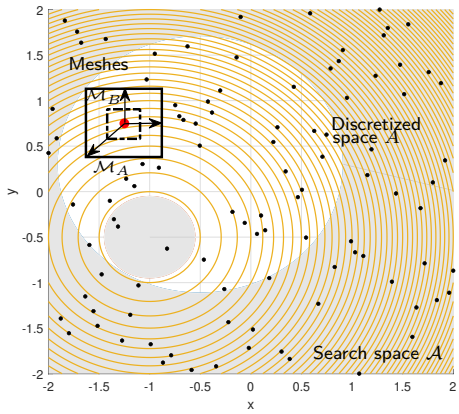
$$w_{\max} := \max\Big\{\max_{x\in G_n} w(x), \max_{x\in M_n} w(x)\Big\}. \tag{14}$$

where $w(x) := \max_i w_n(x,i)$.

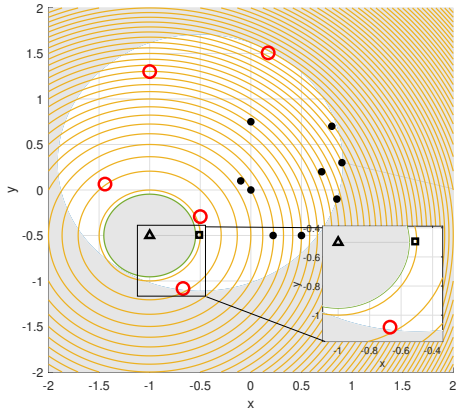
3.1 Reformulation of the maximizers

From the definition of the safe set from (5), we obtain that:

$$x \in S_n \iff x \in \mathcal{A} \text{ and } \forall j = 1,\dots,J \quad l_n(x,j) \geq J_{\min}. \tag{15}$$



(a) Illustration of the discretized search space (black dots) and the meshes used by pattern search with the pattern $[0,1],[1,0],[-1,-1]$ (arrows) around the current point (red circle)



(b) The feasible set is inside the white area. The unconstrained maximum of the quadratic function (31a) is denoted with a triangle and the maximum is indicated with a square. The algorithm used the recommended points (solid circles) starting from the safe set (black dots)

Fig. 1. Illustration of (31)

From the definition of the maximisers (6) we obtain:

$$x \in M_n \iff x \in S_n \text{ and } u_n(x,0) \geq l^* \tag{16}$$

where

$$l^* = \max_z l_n(z,0) \tag{17a}$$

$$\text{subject to } l_n(z,j) \geq J_{\min}, \ \forall j = 1,\dots,J. \tag{17b}$$

We note in (12) that $w_n(\cdot,i)$, $w_n(\cdot,j)$ are independent from each other for $i \neq j$. Therefore, the objective function (12) can be reformulated yielding J separate problems P_1^k, $k = 1,2,\dots,J$. Using (15) and (16), we obtain:

$$P_1^k: \quad \max_{x\in\mathcal{A}} \quad w_n(x,k) \tag{18a}$$

$$\text{subject to} \quad l_n(x,j) \geq J_{\min}, \ \forall j = 1,\dots,J, \tag{18b}$$

$$u_n(x,0) \geq l^*, \tag{18c}$$

where l^* is obtained from (17). The problem in (17) is independent of x and can be solved separately.

Let us denote a solution to the problem P_1^k as x_1^{k*}. Then the solution to (12) is found as:

$$x_1^* = \max_{k=1,\dots,J} x_1^{k*}, \tag{19}$$

obtained for k_1^*.

3.2 Reformulation of the expanders

From (7) we get:

$$\max_{x} \quad \max_{i} w_n(x,i) \tag{20a}$$
$$\text{subject to} \quad x \in S_n, \tag{20b}$$
$$|\mathcal{G}(x)| > 0 \tag{20c}$$

where $\mathcal{G}(\cdot)$ is given by (8). From (8) we notice that $|\mathcal{G}(x)| > 0$ if there exists at least one point $x' \in \mathcal{A} \setminus S_n$ such that the condition:

$$\forall j \; l_{n,j,(x,u_n(x,j))}(x') \geq J_{\min} \tag{21}$$

is satisfied. Thus, we obtain:

$$\max_{x,x'} \; \max_{i} w_n(x,i) \tag{22a}$$
$$\text{subject to } x \in S_n, \tag{22b}$$
$$l_{n,j,(x,u_n(x,j))}(x') \geq J_{\min}, \; \forall j = 1,\dots,J, \tag{22c}$$
$$x' \in \mathcal{A} \setminus S_n. \tag{22d}$$

From the definition of the safe set (5), we get that:

$$x' \in \mathcal{A} \setminus S_n \iff x' \in \mathcal{A} \text{ and } \exists k : l_n(x',k) < J_{\min} \tag{23}$$

Then we have the following equivalence:

$$\exists k : l_n(x',k) < J_{\min} \iff \min_s l_n(x',s) < J_{\min} \tag{24}$$

Then we obtain:

$$\max_{x,x'} \quad \max_{i} w_n(x,i) \tag{25a}$$
$$\text{subject to} \quad \min_s l_n(x,s) \geq J_{\min}, \tag{25b}$$
$$\min_s l_{n,s,(x,u_n(x,s))}(x') \geq J_{\min}, \tag{25c}$$
$$\min_s l_n(x',s) < J_{\min}. \tag{25d}$$

If there exists a solution to (25), the optimization problem can be solved using a derivative-free method, such as pattern search. However, the problem (25) may be infeasible if the set of expanders G_n is empty. To avoid potential infeasibility, we relax (25) as:

$$\max_{x,x'} \quad q(x,x') \tag{26a}$$
$$\text{subject to} \quad \min_s l_n(x,s) \geq J_{\min}, \tag{26b}$$
$$\min_s l_n(x',s) < J_{\min} \tag{26c}$$

where

$$q(x,x') = \max_i w_n(x,i) + \sigma \min\{0, \min_s l_{n,s,(x,u_n(x,s))}(x') - J_{\min}\}$$

where $\sigma > 0$ enables trading off feasibility and optimality. The problem from (26) is feasible if $S_n \subset \mathcal{A}$ in the strict sense, i.e. $S_n \neq \mathcal{A}$. Detecting infeasibility is out of the scope of most derivative-free solvers so we ensured that the problem is feasible in the relaxation (26).

Doing the same reformulation as in (18), we get:

$$P_2^k: \quad \max_{x,x' \in \mathcal{A}} \quad q_k(x,x') \tag{27a}$$
$$\text{subject to} \quad \min_s l_n(x,s) \geq J_{\min}, \tag{27b}$$
$$\min_s l_n(x',s) < J_{\min} \tag{27c}$$

Algorithm 1: Reformulated SafeOpt

Input: Initial safe set $S_0 = \{x^0, x^1, \dots, x^K\} \subset \mathcal{A}$, desired tolerances ϵ_1, ϵ_2, maximal number of iterations M, desired confidence limit α, desired safety threshold $J_{\min}$

Output: Optimal solution x^*

1 Set $n \leftarrow 1$, compute $F_n = \{f(x^i)\}_{i=1,\dots,K}$, $G_{n,j} = \{g_j(x^i)\}_{i=1,\dots,K}$ for $j = 1,\dots,J$, set $S_n \leftarrow S_0$.
2 **repeat**
3 Using S_n and F_n, find a Gaussian process GP_f with lower bounds $l_n(x,0)$, upper bounds $u_n(x,0)$,
4 Using S_n and $G_{n,j}$ find J Gaussian processes $GP_{g,j}$ with lower bounds $l_n(x,j)$, upper bounds $u_n(x,j)$
5 Solve (10), obtaining x_n^* and $l^* = l_n(x_n^*,0)$
6 Solve P_1^j for all $j = 1,\dots,J$ from (18)
7 Solve P_2^j for all $j = 1,\dots,J$ from (27)
8 **if** $q_j(x_2^{j*}, x'^*) \leq w_n(x_2^{j*}, j)$ **then**
9 Set $x_n^r \leftarrow x_1^*$
10 **else**
11 Solve (30) and set $x_n^r \leftarrow \arg\max_{\{x_1^*, x_2^*\}}\{w_n(x_2^*,k_2^*), w_n(x_1^*,k_1^*)\}$
12 **end**
13 Set $n \leftarrow n+1$, set $F_n \leftarrow F_{n-1} \cup \{f(x_n^r)\}$, $G_{n,j} \leftarrow G_{n-1} \cup \{g_j(x_n^r)\}$ for $j = 1,\dots,J$, set $S_n \leftarrow S_{n-1} \cup \{x_n^r\}$
14 **until** $n \geq M$ & $\|x_n^r - x_{n-1}^r\| \leq \epsilon_1$ & $\|f(x_n^r) - f(x_{n-1}^r)\| \leq \epsilon_2$;

where

$$q_k(x,x') = w_n(x,k) + \sigma \min\{0, \min_s\{l_{n,s,(x,u_n(x,s))}(x') - J_{\min}\}\}. \tag{28}$$

Let us denote a solution to the problem P_2^k as x_2^{k*}. Then the solution to (13) is found as:

$$x_2^* = \arg\max_{k=1,\dots,J} w_n(x_2^{k*}, k). \tag{29}$$

obtained for k_2^*. Then from (14) we get:

$$x_n = \arg\max_{x_1^*, x_2^*} \{w_n(x_2^*,k_2^*), w_n(x_1^*,k_1^*)\}. \tag{30}$$

The reformulation proposed in Sections 3.1 and 3.2 is summarized in Algorithm 1.

We note that the problems (18), (27) use the same definitions of the maximisers and the expanders as Berkenkamp et al. (2016). In particular, the proposed reformulation is independent from the chosen pattern search solver. Thus, the problems (10), (18), (27) (lines 5, 6, and 7 in Algorithm 1) are general and can be solved with other methods. For instance, if the problems are differentiable, the reformulation enables using gradient-based methods.

4. EXAMPLES

This section presents the performance of the proposed reformulation. All tests were performed in Windows 10, using Matlab 2021a on a laptop with an AMD Ryzen 7 PRO 5850U, 8 cores, with 32 GB of RAM.

4.1 Safety with non-convex feasible set

We show the safety of the proposed reformulation in an optimization problem with a non-convex feasible set:

$$\max_{x,y} \quad -(x+1)^2 - (y+0.5)^2 \tag{31a}$$

$$\text{subject to} \quad 2 - (x_1+0.5)^2 - (y-0.3)^2 \geq 0, \tag{31b}$$

$$(x+1)^2 + (y+0.5)^2 - 0.2 \geq 0. \tag{31c}$$

The maximum of (31a) is obtained for $x = -1$, $y = -0.5$, which lies outside the feasible set. A feasible maximum is anywhere on the boundary of the set defined by (31c). We set $\epsilon_1 = \epsilon_2 = 0.001$ and $J_{\min} = 0$. An illustration is shown in Fig. 1b. In particular, the inset in the bottom right corner shows a recommended point close to the boundary of the feasible set.

4.2 Unknown constraints

We apply the proposed reformulation to a problem of tuning a cascade PID controller for ball-screw drive from Khosravi et al. (2020, 2022). The control structure is shown in Fig. 2 and the parameters are the same as used by Khosravi et al. (2022). We want to find a parameter K_p for the position controller $C_p(s)$ and the parameters K_v and K_{vi} for the speed control $C_s(s)$ to minimize the integrated absolute error in the position and the overshoot in the speed:

$$J := \gamma \int_0^{t_f} |P(\tau) - P_s(\tau)| \mathrm{d}\tau + \max_{\tau \in [0, t_f]} S(\tau) \tag{32}$$

with $\gamma = 1000$ putting emphasis on the position tracking the desired reference P_s.

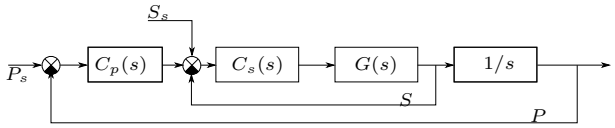


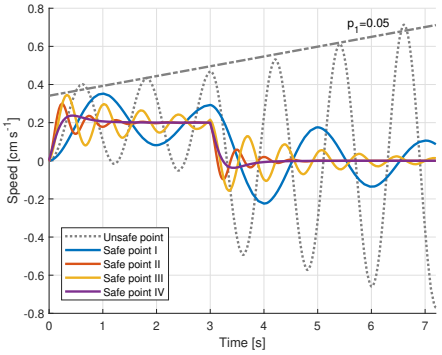
Fig. 2. Block diagram of a ball–screw drive with transfer function $G(s)$. The objective is to follow the position set point P_s ensured by a proportional controller $C_p(s)$ in cascade with a speed controller $C_s(s)$.

To emulate human-driven PID tuning based on visual assessment of responses of the system, we measure stability as the slope p_1 of the peaks of the response of the system, with positive values indicating instability (Åström and Hägglund, 2006, Ch. 4.4). The constraint was formulated as:

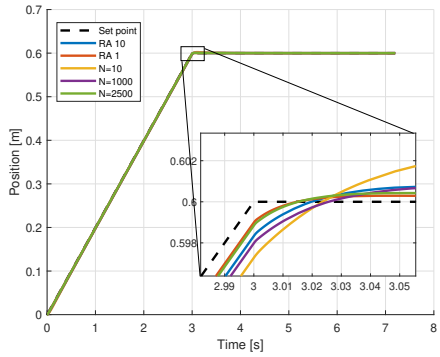
$$h(K_p, K_v, K_{vi}) := p_1 - \sigma \leq 0 \tag{33}$$

where $\sigma = 0.005$ was chosen to ensure that a system with no peaks, i.e. $p_1 = 0$, yields a value inside the feasible set. The grey lines in Fig. 3a show the speed trajectory for an unstable point (dotted line) and the corresponding value p_1 used to evaluate (33). The dash-dotted line with slope p_1 is the linear fit to the peaks of the unstable trajectory.

The search space $\mathcal{A} = [0, 110] \times [0, 50]^2$ was chosen so that it contains unstable values. The initial safe set contains four points found in simulation and the corresponding speed and position are shown in Fig. 3. Setting $x := [K_p, K_v, K_{vi}]^\intercal$, we obtain the problem structure from (1).



(a) Speed for the initial safe set



(b) Position for the solutions

Fig. 3. Speed and position

Performance The performance results are shown in Table 1. The first row shows that strict stopping criteria allowed for reaching a low value of the objective function. However, the overall time needed to get a solution is significant. This is because the algorithm keeps running until the solutions from two consecutive iterations are close. Relaxation of the stopping criteria enabled by the reformulation impacts primarily the number of iterations and thus the overall time to get a solution, reducing it from 268 s (4 min 45 s) to 35 s. Stopping the algorithm after a smaller number of iterations led to a larger value of the objective function, which increased from 3.4 to 6.4. Therefore, we see that in the proposed reformulation we can use the stopping criteria to find a trade-off between the time and the value of the objective function.

Increasing the initial mesh size can bring down the number of iterations (here for 10 and the mesh tolerance equal to 10^{-6}). At the same time, increasing the initial mesh size increases the time to solve (27). This is because pattern search starts looking far from the given initial value. However, as indicated by Kochenderfer and Wheeler (2019), the expanders are supposed to be at the boundary of the safe set, i.e. close to y where $l_n(y, j) = J_{\min}$ for all j. As a result, setting the value of the initial mesh size parameter can serve as passing additional information about the function to the pattern search method. The value can also be problem-dependent, as visible from the values obtained for a small initial mesh size and a large one. A large size of the initial mesh may suggest going outside the search area, in particular, if the current point is close to the boundary of $\mathcal{A}$. This phenomenon can be observed when looking at values of K_{vi} obtained for the initial mesh size of 10, and

Table 1. Performance assessment based on stopping criteria enabled by the reformulation and stopping criteria of pattern search: mesh tolerance ε, initial mesh size δ^0 with the best results marked in bold. An extended version is presented by Zagorowska et al. (2022)

ϵ_1	ϵ_2	ε [$\times 10^{-4}$]	δ^0	K_p^*	K_v^*	K_{vi}^*	Obj. (32)	Iterations	Solution time [s] for Alg. 1	Time [s] for solving (18)	Time [s] for solving (27)
0.001	0.001	0.01	1	107.2	45.3	41.2	-3.4	25	268	0.1	4.2
0.01	0.01	0.01	1	55	31.2	49.3	-6.4	5	44	0.4	6.4
0.1	0.1	0.01	1	55	31.2	49.3	-6.4	4	35	0.6	4.5
0.1	0.1	1	10	60	25.4	50	-5.8	2	13	0.3	4.3
0.1	0.1	100	10	60	25.4	50	-5.8	3	**6**	0.02	0.1
0.1	0.1	100	1	100	43.4	41	**-3.4**	8	17	0.01	0.1

Table 2. Performance of the default SafeOpt

N	Time [s]	Iter.	K_p^*	K_v^*	K_{vi}^*	Obj. Tune
10	4	3	36.7	11.1	27.8	-10.2
1000	14	4	49.7	29.1	43.7	-7
2500	56	5	86.8	34.3	41.8	-4.3

20. In these cases, the solution for K_{vi} obtained from (30) that lies on the boundary. A smaller mesh size, equal to one or five, allowed the solution to lie inside $\mathcal{A}$, resulting in an improved value of the objective.

Comparison with default SafeOpt The performance of the proposed reformulation was compared to the default version of SafeOpt from Section 2.2.1 in terms of the time necessary to obtain a solution. The analysis is collected in Table 2. The time in the second column is the average time obtained from 10 runs of every algorithm. The stopping criteria used for SafeOpt are the number of iterations and evaluation of all the points in the discretized search space. The parameters of the reformulated algorithm were chosen as $\epsilon_1 = \epsilon_2 = 0.1$, with a mesh tolerance of 0.01 and two initial mesh sizes of 10 (RA 10) and one (RA 1). These values were chosen for the shortest solution time and best value of the objective (bold). The results are collected in Table 2 and shown in Fig. 3b. In all the cases, the best objective was obtained for the reformulated algorithm RA 1. The default SafeOpt was second best, at the expense of the computational time.

5. DISCUSSION AND CONCLUSIONS

Existing algorithms for safe learning, such as SafeOpt, allow for ensuring safety at the expense of increased computational effort. The current paper proposes a reformulation of the SafeOpt algorithm as a series of optimization problems. By using direct search methods for the optimization problems, we also preserve the derivative-free character of SafeOpt while improving computation time. In future work, we plan to analyse the impact of the solver used for optimization problems on the convergence of the algorithm and safety guarantees.

REFERENCES

Åström, K.J. and Hägglund, T. (2006). *Advanced PID Control*. ISA-The Instrumentation, Systems, and Automation Society.

Audet, C. and Hare, W. (2017). *Derivative-free and blackbox optimization*. Springer.

Berkenkamp, F., Krause, A., and Schoellig, A.P. (2021). Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *Machine Learning*, 1–35.

Berkenkamp, F., Schoellig, A.P., and Krause, A. (2016). Safe controller optimization for quadrotors with Gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Duivenvoorden, R.R.P.R., Berkenkamp, F., Carion, N., Krause, A., and Schoellig, A.P. (2017). Constrained Bayesian optimization with particle swarms for safe adaptive controller tuning. *IFAC-PapersOnLine*, 50(1), 11800–11807.

Fiducioso, M., Curi, S., Schumacher, B., Gwerder, M., and Krause, A. (2019). Safe contextual Bayesian optimization for sustainable room temperature PID control tuning.

Khosravi, M., Behrunani, V., Smith, R.S., Rupenyan, A., and Lygeros, J. (2020). Cascade control: Data-driven tuning approach based on Bayesian optimization. *IFAC-PapersOnLine*, 53(2), 382–387. 21th IFAC World Congress.

Khosravi, M., Behrunani, V.N., Myszkorowski, P., Smith, R.S., Rupenyan, A., and Lygeros, J. (2022). Performance-driven cascade controller tuning with Bayesian optimization. *IEEE Transactions on Industrial Electronics*, 69(1), 1032–1042.

Kim, Y., Allmendinger, R., and López-Ibáñez, M. (2021). Safe learning and optimization techniques: Towards a survey of the state of the art. In F. Heintz, M. Milano, and B. O'Sullivan (eds.), *Trustworthy AI - Integrating Learning, Optimization and Reasoning*, 123–139. Springer International Publishing, Cham.

Kochenderfer, M.J. and Wheeler, T.A. (2019). *Algorithms for optimization*. MIT Press.

Lewis, R.M. and Torczon, V. (2002). A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4), 1075–1089.

Srinivas, N., Krause, A., Kakade, S.M., and Seeger, M.W. (2012). Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5), 3250–3265.

Sui, Y., Gotovos, A., Burdick, J., and Krause, A. (2015). Safe exploration for optimization with Gaussian processes. In F. Bach and D. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 997–1005. PMLR, Lille, France.

Zagorowska, M., Balta, E.C., Behrunani, V., Rupenyan, A., and Lygeros, J. (2022). Efficient sample selection for safe learning. *arXiv preprint arXiv:2211.14104*.