CONSTRAINED, GLOBAL OPTIMIZATION OF FUNCTIONS WITH LIPSCHITZ CONTINUOUS GRADIENTS

Abraham P. Vinod, Arie Israel, and Ufuk Topcu

ABSTRACT

We present two first-order, sequential optimization algorithms to solve constrained optimization problems. We consider a black-box setting with a priori unknown, non-convex objective and constraint functions that have Lipschitz continuous gradients. The proposed algorithms balance the exploration of the a priori unknown feasible space with the pursuit of global optimality within in a pre-specified finite number of first-order oracle calls. The first algorithm accommodates an infeasible start, and provides either a near-optimal global solution or establishes infeasibility. However, the algorithm may produce infeasible iterates during the search. For a strongly-convex constraint function and a feasible initial solution guess, the second algorithm returns a near-optimal global solution without any constraint violation. In contrast to existing methods, both of the algorithms also compute global suboptimality bounds at every iteration. They can satisfy user-specified tolerances in the computed solution with near-optimal complexity in oracle calls for a large class of optimization problems. We propose tractable implementations of the algorithms by exploiting the structure afforded by the Lipschitz continuous gradient property.

1 Introduction

We study first-order methods to solve the following constrained, global optimization problem,

\[
\begin{align*}
\minimize & \quad J(x), \\
subject \ to & \quad H(x) \leq 0,
\end{align*}
\]

where the functions $J, H : \mathbb{R}^d \rightarrow \mathbb{R}$ are (possibly non-convex) functions with Lipschitz continuous gradients. We consider the black-box setting, where the functions $J$ and $H$ are a priori unknown, and are accessible only via first-order oracles. We denote a global minimum of (1) by $x^\ast$. We propose two sequential optimization algorithms that approximate $x^\ast$ in a finite number of oracle queries. Unlike existing methods, the algorithms provide global suboptimality bounds at every iteration which enable early termination, and the algorithms are worst-case optimal in the budget of the oracle calls required to achieve user-specified tolerances for a large class of problems.

Constrained, global optimization problems of the form (1) are ubiquitous in science and engineering. An application of (1) in machine learning arises in policy optimization for reinforcement learning. Here, we maximize a long-term reward associated with the learning problem by optimizing a parameterized policy, typically a neural network [28]. The constraint $H$ in such problems can impose additional desirable properties or domain-specific knowledge on the policy network. As an illustration, we train a neural network to solve the mountain car problem [3] in Section 4.2, and demonstrate that imposing minimum energy requirements on the closed-loop system allows completion of the task with very few simulations.

A significant part of existing research on global optimization focuses on a special case of (1) [10, 21, 14],

\[
\begin{align*}
\minimize & \quad J(x), \\
subject \ to & \quad x \in \mathcal{X}.
\end{align*}
\]

where $\mathcal{X} \subset \mathbb{R}^d$ is a known, convex, and compact set. A popular approach to tackle (2) in a black-box setting is via iterative optimization of a surrogate optimization problem, constructed using the regularity of $J$ and the information

---

* A. Vinod is with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, 02139, USA. A. Israel is with the Department of Mathematics, and U. Topcu is with the Department of Aerospace Engineering and Engineering Mechanics at the University of Texas at Austin, Austin, TX, 78712 USA; email: aby.vinod@gmail.com, arie@math.utexas.edu, utopcu@utexas.edu. This work was completed while Vinod was a postdoctoral research fellow at the University of Texas at Austin.
from the past oracle queries. For example, see Piyavskii-Shubert algorithm \cite{22, 25} and DIRECT \cite{13} for Lipschitz continuous \( J \), and covering methods \cite{11, 6, 21} for Lipschitz continuous gradients. Alternatively, researchers have utilized hierarchical partitioning of \( \mathcal{X} \) to design optimistic optimization algorithms \cite{17} that solve (2). The Piyavskii-Shubert algorithm, the covering methods, and the optimistic optimization algorithms have deterministic bounds on the global suboptimality for a given budget of oracle calls \cite{11, 17}. On the other hand, random search algorithms utilize Lipschitz information to provide probabilistic budget-dependent bounds on suboptimality \cite{15}. In this paper, we develop novel techniques to perform constrained, global optimization of (1) in a black-box setting, inspired by the covering methods.

Bayesian optimization is another popular sequential optimization approach for black-box optimization \cite{5, 7, 8, 16, 20, 26, 32}. It implicitly imposes regularity requirements on the \textit{a priori} unknown objective and constraint functions by modeling them as samples drawn from a fixed Gaussian processes \cite{16, 24}. At every iteration, it constructs an \textit{acquisition function} using the data from past queries, and solves a surrogate optimization problem to identify the next query point. For the optimization problem (1), existing literature provides budget-dependent, probabilistic-suboptimality bounds on the estimated optimum via regret bounds \cite{3, 27}. For the constrained optimization problem (1), the acquisition function is multiplied with another surrogate function, which models the \textit{probability of feasibility} \cite{5, 7, 8}. To the best of our knowledge, such approaches do not have any convergence guarantees or budget-dependent global suboptimality bounds guarantees. The main advantages of the proposed algorithms presented here over Bayesian optimization techniques are as follows: 1) global suboptimality bounds available from the first feasible iteration, 2) sufficient budgets for the algorithm to achieve user-specified solution tolerances or demonstrate near-infeasibility, and 3) constraint-violation-free optimization of (1), when \( H \) is additionally known to be strongly-convex. Similar to Bayesian optimization problem, the proposed algorithms also solve non-convex surrogate optimization problems. However, due to the structure afforded by Lipschitz gradient continuity, the resulting problems are simpler non-convex, quadratically constrained, quadratic programs, that can be efficiently handled using existing off-the-shelf solvers, like \texttt{GURBO}.

The main contributions of this paper are two first-order, sequential optimization algorithms that approximate the global minimum of (1) with valid global suboptimality bounds under a finite budget of oracle calls. Starting with a (possibly infeasible) initial solution guess, the first algorithm approximates \( x^* \) or proves the (near-)infeasibility of (1). The first algorithm does not require the initial solution guess to be feasible for (1). In contrast, the second algorithm solves (1) without any constraint violation, when the constraint function \( H \) is strongly-convex and the initial solution guess is feasible for (1). Both of the algorithms are \textit{anytime}, i.e., they can be terminated at any point of time to return a valid approximation of \( x^* \) with global suboptimality bound, or a near-infeasibility certificate. We also characterize worst-case, sufficient budgets of oracle calls for the algorithms to achieve a user-specified, global-suboptimality bounds, and show that they are tight up to a constant factor for a large class of problems.

The rest of this paper is organized as follows. Section 2 states the problems of interest, and provides a brief description of mathematical concepts and existing work relevant to solve (1). Section 3 provides the main results of this paper — two algorithms to solve (1) along with the proofs of correctness and a discussion about their implementation. We investigate the efficacy of the proposed algorithms in numerical experiments in Section 4 and conclude in Section 5.

## 2 Setup and preliminaries

We denote the set of natural and real numbers by \( \mathbb{N} \) and \( \mathbb{R} \) respectively, the set of natural numbers (not including zero) by \( \mathbb{N}_+ \), and the set of non-zero natural numbers up to \( t \in \mathbb{N}_+ \) by \( \{1, 2, \ldots, t\} \). For any set \( \mathcal{S} \), \( \mathcal{S}^d \) refers to the Cartesian product of \( \mathcal{S} \) with itself \( d \)-times. We denote the cardinality of a finite set \( \mathcal{S} \) by \( |\mathcal{S}| \), and the absolute value of a scalar \( x \in \mathbb{R} \) by \( |x| \). We use \( \|x\| \) to denote the Euclidean norm of a vector \( x \in \mathbb{R}^d \), and denote the inner between two vectors \( x, y \) by \( x \cdot y \). Given a compact set \( \mathcal{X} \subset \mathbb{R}^d \), we define its diameter as \( \text{diam}(\mathcal{X}) = \sup_{y, x \in \mathcal{X}} \|y - x\| \). The first-order approximation \( \ell : \mathbb{R}^d \rightarrow \mathbb{R} \) of a continuously-differentiable function \( f : \mathcal{X} \rightarrow \mathbb{R} \) about a point \( q \in \mathcal{X} \) is given by

\[
\ell(x; q, f) \triangleq f(q) + \nabla f(q) \cdot (x - q). \tag{3}
\]

Let \( \|\nabla f\|_{\text{max}} \) denote the finite upper bound on \( \|\nabla f\| \) over a compact \( \mathcal{X} \),

\[
\|\nabla f\|_{\text{max}} \triangleq \sup_{x \in \mathcal{X}} \|\nabla f(x)\| < \infty. \tag{4}
\]

We also recall that for any differentiable \( f : \mathcal{X} \rightarrow \mathbb{R} \) and any point \( x, y \in \mathcal{X} \), there exists a point \( z \) on the line joining \( x \) and \( y \), such that

\[
|f(y) - f(x)| = \nabla f(z) \cdot (y - x) \leq \|\nabla f(z)\| \|y - x\| \leq \|\nabla f\|_{\text{max}} \|y - x\|. \tag{5}
\]

Equation (5) follows from mean value theorem and Cauchy-Schwartz inequality.
Lipschitz continuous gradient [18]: Given a set $X \subset \mathbb{R}^d$, a continuously-differentiable function $f : X \to \mathbb{R}$ has a Lipschitz continuous gradient, if its gradient $\nabla f$ satisfies the property $\| \nabla f(y) - \nabla f(x) \| \leq K_f \| y - x \|$ for every $x, y \in X$ for the smallest constant $K_f \in \mathbb{R}$, $K_f \geq 0$. We define a Lipschitz gradient constant $L_f$ as any known upper bound on $K_f$, since $K_f$ is rarely known. We denote the family of functions $f$ with Lipschitz gradient constant $L_f$ by $\mathcal{F}_{L_f}$. For brevity, we will refer to functions with Lipschitz continuous gradients as smooth functions.

Strong-convexity ($\mu$-convexity) [18]: Given a set $X \subset \mathbb{R}^d$, a continuously-differentiable function $f : X \to \mathbb{R}$ is strongly-convex or $\mu$-convex, if for any $x, y \in X$,

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x) + \frac{\mu}{2}\|y - x\|^2,$$

for some convexity constant $\mu > 0$. Similarly to the Lipschitz gradient constant, we do not require $\mu$ to be the largest positive scalar satisfying (6) for every $x, y \in X$. When $f$ is also smooth with Lipschitz gradient constant $L_f$, then $\mu \leq L_f$. We use $\mathcal{F}^\mu_{L_f} \subset \mathcal{F}_{L_f}$ to denote the family of $\mu$-convex, $L_f$-smooth functions.

2.1 Sequential optimization algorithms

Sequential optimization algorithms are popular due to their ease in design, implementation, and analysis. In this paper, we study first-order, sequential optimization algorithms to solve (1).

Definition 1. (FIRST-ORDER, SEQUENTIAL OPTIMIZATION ALGORITHM) Given an initial solution guess $q_1 \in X$ and a budget $T \in \mathbb{N}_+$ of oracle calls, a first-order, sequential optimization algorithm is a procedure that generates a sequence of query points $\{q_t\}_{t=2}^T$ for the first-order oracles for $J$ and $H$. At every iteration $t \in [T-1]$, the algorithm constructs $q_{t+1}$ using the information available until then $\{J(q_k), H(q_k), \nabla J(q_k), \nabla H(q_k)\}$ with $k \in [t]$. The algorithm computes $x^*$ (or an approximation) within $T$ iterations.

Examples of first-order, sequential optimization algorithms include gradient descent and sequential quadratic programming [12, 18, 19].

Unfortunately, due to the richness of the family of smooth functions, even the computation of a feasible solution to (1) using any first-order, sequential optimization algorithm can be arbitrarily difficult under a fixed budget of oracle calls. See Appendix A for such an “adversarial” example. Therefore, we will focus on the computation of an $(\eta^-,-\eta^+,\delta)$-minimum of (1) or proving $\gamma$-infeasibility (near-infeasibility for small $\gamma$).

Definition 2. ($\eta^-,\eta^+,\delta$)-MINIMUM AND $(\eta,\delta)$-MINIMUM OF (1). Given $\eta^-,\eta^+,\delta \geq 0$, a solution $x_{\eta^-,\eta^+,\delta} \in X$ is an $(\eta^-,\eta^+,\delta)$-minimum of (1), provided

$$-\eta^- \leq J(x_{\eta^-,\eta^+,\delta}) - J(x^*) \leq \eta^+ \quad \text{and} \quad H(x_{\eta^-,\eta^+,\delta}) \leq \delta,$$

(7) The solution $x_{\eta^-,\eta^+,\delta}$ is the $(\eta,\delta)$-minimum of (1), when $\eta^- = \eta^+ = \eta$.

Definition 3. ($\gamma$-INEASIBILITY OF (1)). We declare (1) to be $\gamma$-infeasible for some $\gamma \geq 0$, when the following optimization is infeasible,

minimize $J(x)$ \quad subject to $H(x) < -\gamma$.

(8)

2.2 Problem statements

To ensure that (1) does not have an unbounded solution, we make the following standing assumption throughout the paper.

Assumption 1. (FEASIBLE SPACE OF (1) LIES INSIDE A KNOWN, CONVEX AND COMPACT SET). We assume the knowledge of a convex and compact set $X \neq \emptyset$ that contains the a priori unknown feasible set $\{H \leq 0\}$ of (1). When the constraint set $\{H \leq 0\}$ is unbounded, we will seek the (local) minimum of (1) inside the set $X \cap \{H \leq 0\}$.

Apart from the knowledge of $X$, we will assume access to the following to solve (1): 1) the first-order oracles for the $a$ priori unknown functions $J$ and $H$ that provide $(J(q), \nabla J(q))$ and $(H(q), \nabla H(q))$ at any query point $q \in X$ respectively, 2) Lipschitz gradient constants $L_J$ and $L_H$ for $J$ and $H$ over $X$ respectively, and 3) an initial solution guess $q_1 \in X$ that may be infeasible for (1). We now state the two problems of interest.

Problem A. (GLOBAL OPTIMIZATION FOR SMOOTH $H$). Given a budget $T \in \mathbb{N}_+$ of oracle calls and a relaxation threshold $\delta > 0$, design a first-order, sequential optimization algorithm that either declares (1) to be $\gamma$-infeasible for some $\gamma > 0$, or computes a $(\Delta_{\text{global}},\delta)$-minimum of (1) for some $\Delta_{\text{global}} > 0$. Also, given a global-suboptimality threshold $\eta > 0$, characterize the budget $T_{\text{efficient}}$ of oracle calls needed by the algorithm to compute an $(\eta,\delta)$-minimum (when it exists) or declare (1) to be infeasible, irrespective of the choice of $J$, $H$, and $q_1$. 


We obtain the data-driven majorant where the a priori unknown constraint function $H$ is strongly-convex, and the initial solution guess $q_1 \in X$ is feasible for (1). Here, we assume that $H \in \mathcal{F}_{L_H}^\mu$, with known constants $L_H \geq \mu > 0$. Problem $\mathbf{B}$ searches for the global minimum of (1), without violating the a priori unknown constraint $H \leq 0$ in (1).

**Problem B (GLOBAL OPTIMIZATION FOR SMOOTH, STRONGLY-CONVEX $H$).** Given a budget $T \in \mathbb{N}_+$ of oracle calls, $H \in \mathcal{F}_{L_H}^\mu$, and a feasible initial solution guess, design a first-order, sequential optimization algorithm that computes a $(0, \Delta_{\text{global}}, 0)$-minimum of (1) for some $\Delta_{\text{global}} > 0$ without violating the constraint $H \leq 0$ at any iteration. Also, characterize the budget $T_{\text{sufficient}, \mu}$-convex of oracle calls needed by the algorithm to compute an $(0, \eta, 0)$-minimum, irrespective of the choice of $J$, $H$, and $q_1$.

### 2.3 Data-driven approximants for smooth functions

For any function $f : X \to \mathbb{R}$, we define its minorants and majorants as functions $f^-, f^+ : X \to \mathbb{R}$ respectively,

$$f^-(x) \leq f(x) \leq f^+(x), \quad \forall x \in X. \tag{9}$$

Lemma 1 constructs majorants and minorants of a function $f \in \mathcal{F}_{L_f}$ using data as shown in Figure 1.

**Lemma 1 (MAJORANT AND MINORANT FOR $f$).** Consider a function $f : X \to \mathbb{R}$, where $f \in \mathcal{F}_{L_f}$. Given $t \in \mathbb{N}_+$ and data $\{(q_i, f(q_i), \nabla f(q_i))\}_{i=1}^t$, a majorant and minorant of $f$ is given by,

$$f^+_t(x) = \min_{i \in [t]} \left( \ell(x; q_i, f) + \frac{L_f}{2} \|x - q_i\|^2 \right),$$

and

$$f^-_t(x) = \max_{i \in [t]} \left( \ell(x; q_i, f) - \frac{L_f}{2} \|x - q_i\|^2 \right), \tag{11}$$

respectively. Furthermore, $f^+_t(q_i) = f^-_t(q_i) = f(q)$ for every $i \in [t]$, and the approximation errors $f^+_t(x) - f(x)$ and $f(x) - f^-_t(x)$ lie in a bounded interval $[0, \min_{i \in [t]} L_f \|x - q_i\|^2]$.

**Proof.** For any smooth function $f$, the following inequalities hold for any $x \in X$,

$$f(x) \leq \ell(x; q_i, f) + \frac{L_f}{2} \|x - q_i\|^2, \quad \forall i \in [t], \tag{12a}$$

$$f(x) \geq \ell(x; q_i, f) - \frac{L_f}{2} \|x - q_i\|^2, \quad \forall i \in [t]. \tag{12b}$$

We obtain the data-driven majorant $f^+_t$ [10] and minorant $f^-_t$ [11] via finite minimum of (12a) and finite maximum of (12b) over $i \in [t]$ respectively. By construction, these piecewise-quadratic functions coincide with $f(q_i)$ at $x = q_i$ for every $i \in [t]$. Also,

$$(12a) \Rightarrow f(x) - \left( \ell(x; q_i, f) - \frac{L_f}{2} \|x - q_i\|^2 \right) \leq L_f \|x - q_i\|^2, \tag{13}$$

$$(12b) \Rightarrow \left( \ell(x; q_i, f) + \frac{L_f}{2} \|x - q_i\|^2 \right) - f(x) \leq L_f \|x - q_i\|^2. \tag{14}$$

We obtain an upper bound on the approximation errors $f - f^-_t$ and $f^+_t - f$ by computing the finite minimum of (13) and (14) over $i \in [t]$.

Using (12), a $\mu$-convex $f$ admits a tighter, piecewise-quadratic, data-driven minorant $f^-_t, \mu : X \to \mathbb{R}$,

$$f^-_{t, \mu}(x) = \max_{i \in [t]} \left( \ell(x; q_i, f) + \frac{\mu}{2} \|x - q_i\|^2 \right), \tag{15}$$

with $f^-_t \leq f^-_{t, \mu} \leq f$. 

---

**Figure 1:** Approximants for $f \in \mathcal{F}_{L_f}$
Algorithm 1: Covering method for global optimization of \((\text{2})\)

**Input:** Convex & compact set \(\mathcal{X} \subset \mathbb{R}^d\), first-order oracle for \(J\), initial point \(q_1 \in \mathcal{X}\), Lipschitz gradient constant \(L_J\), suboptimality threshold \(\eta > 0\)

**Output:** Near-global minima of \((\text{2})\) with suboptimality bound

1. Initialize \(x_{\text{global}}^1 \leftarrow q_1\) and \(\Delta_{\text{global}} \leftarrow \infty\), and query the first-order oracle at \(q_1\)
2. for \(t = 1, 2, 3, \ldots\) do
   3. Construct \(J_t^-\) using \((\text{11})\)
   4. Solve the following optimization problem to compute \(q_{t+1}\) and \(J_t^- (q_{t+1})\)
      \[
      q_{t+1} \leftarrow \underset{x}{\arg\inf} \ J_t^- (x) \text{ subject to } x \in \mathcal{X}
      \]
   5. Query the first-order oracle at \(q_{t+1}\)
   6. Update the near-global minima estimate and suboptimality bound:
      \[
      x_{\text{global}}^t \leftarrow \underset{q_i: i \in [t+1]}{\arg\min} J(q_i)
      \]
      \[
      \Delta_{\text{global}} \leftarrow \min_{q_i: i \in [t+1]} (J(q_i) - J_t^- (q_{t+1}))
      \]
7. if \(\Delta_{\text{global}} \leq \eta\) then break \(\triangleright\) Terminate, if acceptable global suboptimality bound
8. return \((x_{\text{global}}^t, \Delta_{\text{global}})\) \(\triangleright\) \(\Delta_{\text{global}} < \infty\) at the end of first iteration

2.4 Covering method for global optimization of \((\text{2})\)

We briefly discuss how covering method solves \(\inf_{x \in \mathcal{X}} J(x)\), which motivates the proposed algorithms. Algorithm \((\text{11})\) computes \(x_{\text{global}}^1 \in \mathcal{X}\) and \(\Delta_{\text{global}} \geq 0\) such that

\[
0 \leq J(x_{\text{global}}^1) - J(x^*) \leq \Delta_{\text{global}}.
\]

At each iteration of Algorithm \((\text{11})\) the optimization problem \((\text{16})\) is feasible and has a finite optimal solution since \(\mathcal{X}\) is compact and non-empty. Furthermore, \(x_{\text{global}}^t\) satisfies \((\text{19})\) at every iteration \(t \in \mathbb{N}_+\), since

\[
J_t^- (q_{t+1}) \leq J_t^- (x^*) \leq J(x^*) \leq J(x_{\text{global}}^t) \triangleq \min_{i \in [t+1]} J(q_i).
\]

Equation \((\text{20})\) follows from \((\text{11}), (\text{16})\), and the fact that \(x_{\text{global}}^t\) is always feasible for \((\text{2})\). See \((\text{1}), (\text{6}), (\text{21})\) for more details.

To characterize the upper limit on the number of iterations required to ensure that the global-suboptimality bound is below \(\eta\), we first recall Lemma \((\text{2})\) which follows from the pigeonhole principle.

**Lemma 2.** For any \(\epsilon > 0\), any convex and compact set \(\mathcal{X} \subset \mathbb{R}^d\), and any finite collection of \(T \geq \left\lceil \left( \frac{\text{diam}(\mathcal{X}) \sqrt{d}}{\epsilon} \right)^d \right\rceil + 1\) distinct points \(q_i \in \mathcal{X}\) for every \(i \in [T]\), there exists \(t \in [T-1]\) such that

\[
\min_{i \in [t]} \|q_{t+1} - q_i\|^2 \leq \epsilon.
\]

**Proof.** The set \(\mathcal{X}\) is covered by a hypercube of side \(\text{diam}(\mathcal{X})\). For any \(\epsilon > 0\) The minimum number of hypercubes of side \(\sqrt{\epsilon}\) that covers the hypercube of side \(\text{diam}(\mathcal{X})\) is given by \(\left\lceil \frac{\text{diam}(\mathcal{X})^d}{\epsilon^d} \right\rceil\). Note that \(T\) is at least one more than this minimum number. By the pigeonhole principle, at least one of the hypercubes with side \(\sqrt{\epsilon}\) must have at least two points. However, the maximum separation allowed between two points within such a hypercube is \(\sqrt{\epsilon}\). Thus, for some \(i, j \in [T]\) with \(i \neq j\), we have \(\|q_i - q_j\| \leq \sqrt{\epsilon}\). We complete the proof with \(t \triangleq \max(i, j) - 1 \in [T-1]\).

**Proposition 1 (Worst-Case, Sufficient Budget for Algorithm \((\text{1})\)).** For a user-specified suboptimality threshold \(\eta > 0\), define \(T = \left\lceil \left( \frac{\text{diam}(\mathcal{X}) \sqrt{d}}{\epsilon} \right)^d \left( \frac{L_J}{\eta} \right)^2 \right\rceil + 1\). Algorithm \((\text{1})\) terminates at an iteration \(t \leq T\) satisfying \((\text{19})\) with \(\Delta_{\text{global}} \leq \eta\), irrespective of the choice of \(J \in \mathcal{F}_{L_J}\) and the initial solution \(q_1 \in \mathcal{X}\).

**Proof.** From \((\text{18})\) and Lemma \((\text{2})\) we have for every \(t \in [T-1]\),

\[
\Delta_{\text{global}} = J(x_{\text{global}}^t) - J_t^- (q_{t+1}) \leq J(q_{t+1}) - J_t^- (q_{t+1}) \leq L_J \min_{i \in [t]} \|q_{t+1} - q_i\|^2.
\]
We now present the main results, and address Problems A and B. Specifically, we propose Algorithms 2 and 3 (see page 7) for the global optimization of (1). These first-order sequential optimization algorithms solve tractable, surrogate optimization problems of (1), constructed using the past oracle queries and smoothness information, to determine the next query point. Algorithm 3 achieves constraint violation-free optimization by performing an additional projection step.

To help the reader put the proposed algorithms in context, we provide two illustrative examples in page 8. The first example demonstrates how the choice of hyperparameter \( L_j \) in Algorithm 2 affects the approximants, and consequently, the number of iterations to solve (1) and the number of constraint violations. The second example shows that Algorithm 3 can compute a solution to (1), without incurring any constraint violation, when \( H \) is additionally known to be strongly-convex. Both of the algorithms escape a local minimum near the initial solution guess \( q_1 \) to arrive at the global minimum. We provide the numerical details of the examples in Appendix B.

3 Tractable algorithms for global optimization of (1)

We now present the main results, and address Problems A and B. Specifically, we propose Algorithms 2 and 3 (see page 7) for the global optimization of (1). These first-order sequential optimization algorithms solve tractable, surrogate optimization problems of (1), constructed using the past oracle queries and smoothness information, to determine the next query point. Algorithm 3 achieves constraint violation-free optimization by performing an additional projection step.

To help the reader put the proposed algorithms in context, we provide two illustrative examples in page 8. The first example demonstrates how the choice of hyperparameter \( L_j \) in Algorithm 2 affects the approximants, and consequently, the number of iterations to solve (1) and the number of constraint violations. The second example shows that Algorithm 3 can compute a solution to (1), without incurring any constraint violation, when \( H \) is additionally known to be strongly-convex. Both of the algorithms escape a local minimum near the initial solution guess \( q_1 \) to arrive at the global minimum. We provide the numerical details of the examples in Appendix B.

3.1 Global optimization of (1) for smooth (possibly non-convex) \( H \)

Algorithm 2 solves (1) with smooth (possibly non-convex) functions \( J \) and \( H \). It constructs iterates by solving a surrogate optimization problem (23). Note that every feasible solution of (1) is feasible for (23), since the constraint \( H \leq 0 \) is a relaxation of the constraint \( H \leq 0 \) in (1) (Lemma 1). Motivated by Algorithm 1, Algorithm 2 also replaces the unknown objective \( J \) of (1) with its known, data-driven minorant \( J_t \) in (23). Algorithm 2 does not require a feasible initial solution guess \( q_1 \) to solve (1).

We prove the correctness of Algorithm 2 using a \( \delta \)-approximation of (1).

\[
\text{\text{\( \delta \)-relax of (1)}: \quad \text{minimize } J(x) \quad \text{subject to } x \in \mathcal{X}, \ H(x) \leq \delta. \quad (22)\]

The optimal values of (1) and (22) are closely related under the following assumption.

**Assumption 2 (Well-behaved \( H \) at the boundary of \( \{H \leq 0\} \)).** The constraint function \( H \) and the relaxation threshold \( \delta > 0 \) satisfies \( \|\nabla H(x)\| \geq \sqrt{2L_H \delta} \) for every \( x \in \{0 < H \leq \delta\} \).

Assumption 2 ensures that the gradient of \( H \) over the “excess” feasible space, the set \( \{0 < H \leq \delta\} \) arising from relaxation of the constraint \( \{H \leq 0\} \) to \( \{H \leq \delta\} \) is bounded away from zero (Figure 2 on page 9). The requirement on \( H \) imposed by Assumption 2 weakens as the user-specified relaxation threshold \( \delta \) approaches zero.

**Proposition 2 (\( \delta \)-Relaxation (22) Closely Approximates (1)).** Let Assumption 2 hold. Then, whenever (1) is feasible, the global minimum \( x^*_1 \in \mathcal{X} \) of (22) is related to the global minimum \( x^* \) of (1), in the following sense

\[
0 \leq J(x^*) - J(x^*_1) \leq \|\nabla J\|_{\max} \sqrt{\frac{2\delta}{L_H}}. \quad (30)
\]

**Proof.** Note that the feasible set \( \mathcal{X} \cap \{H \leq \delta\} \) of (22) is non-empty and bounded, and contains the non-empty feasible solution space of (1). Therefore, \( x^*_1 \) exists, and we trivially have the lower bound \( J(x^*) - J(x^*_1) \geq 0 \).

To prove the proposed upper bound on \( J(x^*) - J(x^*_1) \), we study two cases — \( H(x^*_1) \leq 0 \) and \( 0 < H(x^*_1) \leq \delta \). In either cases, we will construct a feasible solution \( z \) for (1), and characterize an upper bound for \( J(z) \) using \( J(x^*_1) \) to complete the proof. For the first case \( (H(x^*_1) \leq 0) \), we choose \( z = x^*_1 \) to trivially satisfy the upper bound.
**Common inputs:** Convex & compact set $\mathcal{X} \subset \mathbb{R}^d$ that contains $\{H \leq 0\}$, first-order oracles for $J$ and $H$, initial solution guess $q_1 \in \mathcal{X}$, suboptimality threshold $\eta > 0$, Lipschitz gradient constants $L_J \geq 0$ and $L_H \geq 0$, budget of oracle queries $T \in \mathbb{N}_+$.

**Algorithm 2** Global optimization of (1) for smooth (possibly non-convex) $H$

**Other requirements:** Relaxation threshold $\delta > 0$, $(H, \delta)$ that satisfies Assumption 2

**Output:** Near-global minima with suboptimality bound or prove (near-)infeasibility

1. Initialize $x^*_{\text{global}} \leftarrow -\infty$ and $\Delta_{\text{global}} \leftarrow -\infty$, and query the first-order oracles at $q_1$

2. for $t \in [T - 1]$ do
   3. Construct $J^-_t$ and $H^-_t$ using (11)
   4. Solve the following optimization problem to compute $q_{t+1}$ and $J^-_t(q_{t+1})$
      \[ q_{t+1} \leftarrow \arg \inf_x J^-_t(x) \quad \text{subject to} \quad x \in \mathcal{X}, H^-_t(x) \leq 0. \]  
   5. if (23) is infeasible then return (1) is infeasible
   6. else
      7. Query first-order oracles at $q_{t+1}$
         8. if $H(q_{t+1}) \leq \delta$ then
            9. Update the near-global minima estimate and suboptimality bound:
               \[ x^*_{\text{global}} \leftarrow \arg \min_{\{q_i : i \in [t+1], H(q_i) \leq \delta\}} J(q_i) \]  
               \[ \Delta_{\text{global}} \leftarrow \min_{\{q_i : i \in [t+1], H(q_i) \leq \delta\}} J(q_i) - J^-_t(q_{t+1}) \]  
         10. if $\Delta_{\text{global}} \leq \eta$ then break
            > Terminate early, if acceptable global suboptimality
      11. if $\Delta_{\text{global}}$ is $\infty$ then return (1) is $\gamma$-infeasible with $\gamma \leftarrow -\min_{x \in \mathcal{X}} H^-_t(x), 0$
      12. else return $(x^*_{\text{global}}, \Delta_{\text{global}})$

**Algorithm 3** Global optimization of (1) for smooth, strongly-convex $H$

**Other requirements:** Convexity constant $0 < \mu \leq L_H$ for $H$, $q_1$ is feasible for (1)

**Output:** Near-global minima with suboptimality bound using only feasible queries

1. Initialize $x^*_{\text{global}} \leftarrow q_1$ and $\Delta_{\text{global}} \leftarrow -\infty$, and query the first-order oracles at $q_1$

2. for $t \in [T - 1]$ do
   3. Construct $J^-_t$, $H^+_t$, and $H^-_{t, \mu}$ using (11), (10), and (15) respectively
   4. Solve the following optimization problems to compute $\xi_{t+1}$, $J^-_t(\xi_{t+1})$, and $q_{t+1}$:
      \[ \text{Relax} : \quad \xi_{t+1} \leftarrow \arg \inf_x J^-_t(x) \quad \text{subject to} \quad x \in \mathcal{X}, H^-_{t, \mu}(x) \leq 0, \]  
      \[ \text{Project} : \quad q_{t+1} \leftarrow \arg \inf_x \|x - \xi_{t+1}\| \quad \text{subject to} \quad x \in \mathcal{X}, H^+_t(x) \leq 0. \]  
   5. Query the first-order oracles at $q_{t+1}$
   6. Update the near-global minima estimate and suboptimality bound:
      \[ x^*_{\text{global}} \leftarrow \arg \min_{\{q_i : i \in [t+1]\}} J(q_i) \]  
      \[ \Delta_{\text{global}} \leftarrow \min_{\{q_i : i \in [t+1]\}} J(q_i) - J^-_t(\xi_{t+1}) \]  
   7. if $\Delta_{\text{global}} \leq \eta$ then break
      > Terminate early, if acceptable global suboptimality
   8. return $(x^*_{\text{global}}, \Delta_{\text{global}})$
      > $\Delta_{\text{global}} < \infty$ at the end of first iteration
Example 1: Non-convex $H$, infeasible start

Algorithm 2 with $L_J = 0.2$ computes an $(0, \eta, \delta)$-minimum in 14 iterations with 4 infeasible queries.

Algorithm 2 with $L_J = 5 \times 0.2$ computes an $(0, \eta, \delta)$-minimum in 28 iterations with 7 infeasible queries.

Example 2: $\mu$-convex $H$, feasible start

Algorithm 2 computes an $(0, \eta, \delta)$-minimum in 13 iterations with 4 infeasible queries.

Algorithm 3 computes an $(0, \eta, 0)$-minimum in 14 iterations with 0 infeasible queries.

Table 1: Illustration of Algorithms 2 and 3 with $\eta = 0.01$ and $\delta = 10^{-5}$. (top-left and top-right) Algorithm 2 needs more iterations to solve (1) for larger $L_J$, while handling non-convex $H$ and infeasible start. (bottom-left and bottom-right) For a $\mu$-convex $H$ and feasible start, Algorithm 3 computes near-global optimum without any infeasible queries. Algorithm 2 constructs a monotone decreasing sequence of $\{H_t \leq 0\}$ such that $x^* \in \{H_t \leq 0\}$ at every iteration, while Algorithm 3 constructs a monotone increasing sequence of $\{H_t \leq 0\}$ such that $x^* \in \{H_t \leq 0\}$ at some iteration. Legend: initial guess $q_1$ (♦), feasible (+) and infeasible (x) queries for (1).
Theorem 1 (Algorithm 2 addresses Problem [A].) For any objective function $J$, constraint function $H$, and initial solution $q_1$, the following statements about Algorithm 2 are true for any relaxation threshold $\delta > 0$.

A. (Well-definedness) For any iteration of Algorithm 2, the optimization problem (23) either has a finite optimal solution or generates a proof of infeasibility of (1) (resulting in the termination of Algorithm 2 at Step 5).

B. (Anytime Property) Let Algorithm 2 run up to an iteration $t \in \mathbb{N}$, without arriving at a proof of infeasibility of (1) (Theorem 3).

(a) If $\Delta_{\text{global}} < \infty$, then $x^*_\text{global}$ is a $\left(\left\|\nabla J\right\|_{\text{max}} \sqrt{\frac{2\delta}{L_H}}, \Delta_{\text{global}}, \delta\right)$-minimum of (1). Additionally, if $x^*_\text{global}$ is feasible for (1), then $x^*_\text{global}$ is a $(0, \Delta_{\text{global}}, \delta)$-minimum of (1).

(b) Else ($\Delta_{\text{global}} = \infty$), (1) is $\gamma$-infeasible, $\gamma \triangleq -\min_{x \in X} H^-_{\gamma}(x), 0$.

C. (Worst-case, Sufficient Budget) For a suboptimality threshold $\eta \geq \left\|\nabla J\right\|_{\text{max}} \sqrt{\frac{2\delta}{L_H}}$, define

$$T_{\text{sufficient}} \triangleq \left\lfloor \text{(diam}(X)\sqrt{\delta})^d \left(\frac{L_H}{\eta} \right)^{\frac{d}{2}} + \left(\frac{L_H}{\delta}\right)^{\frac{d}{2}} \right\rfloor + 1.$$

Then, Algorithm 2 terminates with an $(\eta, \delta)$-minimum of (1) or declares that (1) is infeasible at some iteration $t \leq T_{\text{sufficient}}$.

Proof. Proof of (A) By Lemma 1 every feasible solution of (1) is feasible for (23) at every iteration of Algorithm 2 since $H_{t-1} \leq H$ for every iteration $t$. Therefore, the infeasibility of (23) at any iteration implies the infeasibility of (1). On the other hand, due to the compactness of $X$, the optimal solution of (23) is finite whenever (23) is feasible.

Proof of (B) Since $\Delta_{\text{global}} < \infty$ at the end of iteration $t \in [T - 1]$, we know that there is some iteration $i \in [t]$ such that $H(q_{i+1}) \leq \delta$. Consequently, $x^*_\text{global}$ is well-defined, and $H(x^*_\text{global}) \leq \delta$. From (22) and (24), $J(x^*_\text{global}) \leq J(x^*_\text{global}) + \delta$.
We now briefly discuss a minor modification to Algorithm 2 that can significantly improve the computed near-suboptimality \( J(q_t^+) \). We define \( \gamma = -\min_{H \in \mathcal{X}} H(x) \), which does not emphasize on the value of \( H \) until either \( H(q_t^+) \leq \delta \), which ensures that \( \gamma \) is feasible for (23)) and \( \Pi \) is non-empty. Therefore, for every iteration \( t \in [T - 1] \). Recall that \( H(q_{t+1}) - H(q_t^+) \leq L_H \min_{i \in [t]} \| q_{t+1} - q_i \|^2 \) at every iteration \( t \in [T - 1] \) by Lemma 1. We obtain a contradiction of (32), as desired, via Lemma 2 and the choice of \( T_\delta \).

Having ruled out the second outcome for \( T = T_{\text{sufficient}} > T_\delta \), we now turn to the third outcome. We claim that, in this case, Algorithm 2 terminates with an \((\eta, \delta)\)-minimum of (11) at some iteration \( t \in [T_{\text{sufficient}} - 1] \). To prove the claim, we again pursue a proof via contradiction. Specifically, we assume for contradiction that (23) is feasible at every iteration \( t \in [T_{\text{sufficient}} - 1] \), the set \( \Pi \) is non-empty, and \( \Delta_{\text{global}} > \eta \). By the same arguments used to rule out the second outcome, \( \Pi \geq T_{\text{sufficient}} - (T_\delta - 1) \). From Lemma 1, we arrive at the contradiction that for every \( t \in \Pi \),

\[
\Delta_{\text{global}} \triangleq J(x^+_{\text{global}}) - J(q^+_{t+1}) \leq J(q_{t+1}) - J(q^+_{t+1}) \leq L_J \min_{i \in [t]} \| q_{t+1} - q_i \|^2 \leq L_J \min_{i \in \Pi} \| q_{t+1} - q_i \|^2 \leq \eta
\]

by Lemma 2 and the choice of \( T_{\text{sufficient}} \).

We now briefly discuss a minor modification to Algorithm 2 that can significantly improve the computed near-infeasibility certificate \( \gamma \). The modification is relevant only when the feasibility of (11) is unknown and the initial solution guess \( q_1 \) is infeasible. Instead of determining \( \gamma \) based on the iterates obtained from solving (23) (Step 5 of Algorithm 2), we design oracle queries based on the following rule

\[
q_{t+1} = \inf_{x \in \mathcal{X}} H(x^+_{t+1}),
\]

until either \( H(q_{t+1}) \leq \delta \), which ensures that \( \Delta_{\text{global}} < \infty \), or we reach the prescribed budget of oracle calls. The rule (33), which is to be executed before Step 2 of Algorithm 2, is motivated by (16) in Algorithm 1. In contrast to (23) which does not emphasize on the value of \( H \), (33) seeks to minimize \( H \) in an effort to find a feasible point for (11). Note that this modification only affects the constants of \( T_{\text{sufficient}} \) prescribed in Theorem 1, thanks to Proposition 1.
3.1.1 Tightness of the worst-case analysis

Lemma 3 recalls a known lower bound on the oracle call complexity for constrained, global optimization [33, Sec. 1.6].

**Lemma 3 (WORST-CASE NECESSARY BUDGET).** For any $\nu > 0$ and $K > 0$, there exist twice-differentiable $J \in \mathcal{F}_K$ and $H \in \mathcal{F}_K$ such that any first-order sequential optimization algorithm takes $\left[ \frac{C}{\nu K \delta^{\frac{1}{2}}} \right]$ oracle calls to compute a $(0, \nu K \delta^{\frac{1}{2}})$-minimum. Here, $C$ is a positive constant that is independent of $\nu$.

Using Lemma 3, we conclude that for any $\eta > 0$, there is a problem instance of (1) with twice-differentiable objective and constraint functions for which Algorithm 2 needs at least $\left[ \frac{C}{(2\nu \eta)^{\frac{1}{2}}} \right] \frac{1}{4} d$ oracle calls to compute a $(0, \eta)$-minimum of (1). On the other hand, even for such an “adversarial” problem instance, Algorithm 3 needs at most $2\left( \sqrt{d} \right)^{\frac{1}{2}} \frac{d}{(2\nu \eta)^{\frac{1}{2}}} + 1$ oracle calls to compute a $(\eta, \eta)$-minimum of (1) by Theorem 1C, provided the user-specified relaxation threshold $\delta \leq \frac{K \eta^2}{2\nu \eta^{\frac{3}{2}}}$. In other words, $T_{\text{sufficient}}$ prescribed for Algorithm 2 is sufficient and necessary (up to constant factors) for a large subclass of problems of the form (1).

3.2 Global optimization of (1) for smooth, strongly-convex $H$ without constraint violation

Algorithm 3 addresses Problem B to compute a near-global minimum for (1) without any constraint violation. It requires the constraint function $H$ be strongly-convex with a known convexity constant $\mu > 0$ and a feasible initial solution guess $x^0$. Unlike Algorithm 2, Algorithm 3 does not require Assumption 2 or the $\delta$-relaxation (22).

Algorithm 3 follows a relax-and-project approach to create a monotonically-decreasing sequence of outer-approximations $\{ H^\mu \}$ and a monotonically-increasing sequence of inner-approximations $\{ H^\mu \}$ of a priori unknown set $H$. Thanks to the $\mu$-convexity of $H$ and a feasible initial solution guess $q_1$, these approximations are non-empty. Algorithm 3 ensures that the queries $q_{t+1}$ are feasible for (1) for every $t \in [T - 1]$ via a projection step (27). Consequently, Algorithm 3 computes a $(0, \Delta_{\text{global}}, 0)$-minimum of (1) at every iteration, and accommodates non-convex, smooth objective functions, similar to Algorithm 2.

**Theorem 2 (ALGORITHM 3 ADDRESSES PROBLEM B).** For any objective function $J$, $\mu$-convex constraint function $H$, and feasible initial solution $q_1$, the following statements about Algorithm 3 hold:

A. (WELL-DEFINEDNESS) The optimization problems (26) and (27) in Algorithm 3 are always feasible and have a finite optimal solution at all iterations.

B. (NO CONSTRAINT VIOLATION) All queries of Algorithm 3 are feasible for (1).

C. (ANYTIME PROPERTY) Let Algorithm 2 run up to an iteration $t \in \mathbb{N}$. Then, $x^\dagger_t$ is a $(0, \Delta_{\text{global}}, 0)$-minimum of (1).

D. (WORST-CASE, SUFFICIENT BUDGET) For a suboptimality threshold $\eta > 0$, define $\kappa \triangleq \frac{1}{LJ \left( \frac{LJ \lVert \nabla J \rVert_{\max}}{2LJ \lVert \mu \rVert_{\max}} + \frac{2LJ \mu}{\mu} \right)}$, and

\[
T_{\text{sufficient, } \mu \text{-convex}} \triangleq \left( \text{diam}(X) \sqrt{d} \right)^{\frac{1}{2}} \left( \frac{\kappa}{\eta} \right)^{\frac{1}{2}} + 1. \tag{34}
\]

Then, Algorithm 3 terminates with an $(0, \eta, 0)$-minimum of (1) at some iteration $t \leq T_{\text{sufficient, } \mu \text{-convex}}$.

**Proof.** Proof of A) The optimization problems (26) and (27) always admit $q_1$ as a feasible solution $H(q_1) = H_{-\mu}(q_1) = H^\mu(q_1)$, and therefore have a non-empty feasible solution space. Furthermore, since $X$ is compact, these optimization problems have a well-defined global minima.

Proof of B) The proof of feasibility of $q_{t+1}$ for (1) at every iteration $t \in [T - 1]$ follows from the observation that $H(q_{t+1}) \leq H_{\mu}(q_{t+1}) \leq 0$ by (27) and Lemma 1.

Proof of C) Since $H_{-\mu} \leq H_{\mu}$, (26) is a relaxation of (1), with the constraint $H \leq 0$ relaxed to $H_{-\mu} \leq 0$. Consequently, we have the following inequality at every iteration $t \in [T - 1]$ (similar to (31)),

\[
J_t^\mu(\xi_{t+1}) \leq J_t^\mu(x^*) \leq J(x^*) \leq J(x_{\text{global}}^*) \leq \min_{i \in [t+1]} J(q_i) \leq J(q_{t+1}). \tag{35}
\]
From (35), we have the following bounds on the true global suboptimality,

\[ 0 \leq J(x_{\text{global}}^t) - J(x^*) \leq \Delta_{\text{global}} \triangleq \min_{i \in [t+1]} J(q_i) - J_i^-(\xi_{t+1}). \]

(36)

Thus, \( x_{\text{global}}^t \) is a \((0, \Delta_{\text{global}}, 0)\)-minimum of \( \Pi \) at every iteration since \( H(x_{\text{global}}^t) \leq 0 \) by Theorem 38 and (38).

**Proof of (D)** We seek a lower bound on the budget \( T \) of oracle calls, which ensures \( \Delta_{\text{global}} \leq \eta \). Using mean value theorem, Lemma 1, the definition of \( \|\nabla J\|_{\text{max}} \), and (35), we characterize the following upper bound on \( \Delta_{\text{global}} \).

\[ \Delta_{\text{global}} \leq J(q_{t+1}) - J_i^-(\xi_{t+1}) = J(q_{t+1}) - \max_{i \in [t]} \left( \ell(\xi_{t+1}; q_i, J) - \frac{L_j}{2} \| \xi_{t+1} - q_i \|^2 \right) \]

\[ = J(q_{t+1}) - J(\xi_{t+1}) + \max_{i \in [t]} \left( \ell(\xi_{t+1}; q_i, J) - \frac{L_j}{2} \| \xi_{t+1} - q_i \|^2 \right) \]

\[ \leq \| \nabla J \|_{\text{max}} \| q_{t+1} - \xi_{t+1} \| + L_j \min_{i \in [t]} \| \xi_{t+1} - q_i \|^2. \]

(37)

We will upper bound (37) using \( \min_{i \in [t]} \| q_{t+1} - q_i \|^2 \) to complete the proof using Lemma 2.

For the given budget \( T \), define \( \Pi_T \subseteq [T - 1] \) as the finite set of iterations where \( q_{t+1} = \xi_{t+1} \), i.e., (27) resulted in a trivial projection. At any iteration \( t \), we have two cases — \( t \in \Pi \) or \( t \notin \Pi \). For the first case, we have \( q_{t+1} = \xi_{t+1} \), which implies

\[ \Delta_{\text{global}} \leq L_j \min_{i \in [t]} \| q_{t+1} - q_i \|^2. \]

(38)

We now consider the second case — \( t \notin \Pi \), where (27) generates a non-trivial projection point \( q_{t+1} \neq \xi_{t+1} \). Here, we will upper bound the terms in (37) separately to characterize the sufficient budget. We will show that the proposed upper bound for the second case subsumes (38) to complete the proof using Lemma 2 and (36).

First, we show the following upper bound to the second term in (37),

\[ \min_{i \in [t]} \| \xi_{t+1} - q_i \|^2 \leq \frac{2 \| \nabla H \|_{\text{max}}}{\mu} \| q_{t+1} - \xi_{t+1} \| + \frac{L_H}{\mu} \min_{i \in [t]} \| q_{t+1} - q_i \|^2. \]

(39)

To prove (39), we first recall that \( H_{t,\mu}(\xi_{t+1}) \leq 0 = H_i^+(q_{t+1}) \) from (26) and (27) at every \( t \notin \Pi \). Consequently,

\[ H_{t,\mu}(\xi_{t+1}) \triangleq \max_{i \in [t]} \left( \ell(\xi_{t+1}; q_i, H) + \frac{H}{2} \| \xi_{t+1} - q_i \|^2 \right) \leq H_i^+(q_{t+1}) \triangleq \min_{i \in [t]} \left( \ell(q_{t+1}; q_i, H) + \frac{L_H}{2} \| q_{t+1} - q_i \|^2 \right) \]

\[ \Rightarrow \forall i \in [t], \mu \| \xi_{t+1} - q_i \|^2 \leq 2 \| \nabla H(q_i) \cdot (q_{t+1} - \xi_{t+1}) + L_H \| q_{t+1} - q_i \|^2, \]

\[ \Rightarrow \forall i \in [t], \mu \| \xi_{t+1} - q_i \|^2 \leq 2 \| \nabla H(q_i) \| \| q_{t+1} - \xi_{t+1} \| + L_H \| q_{t+1} - q_i \|^2, \]

\[ \Rightarrow \forall i \in [t], \| \xi_{t+1} - q_i \|^2 \leq \frac{2 \| \nabla H \|_{\text{max}}}{\mu} \| q_{t+1} - \xi_{t+1} \| + \frac{L_H}{\mu} \| q_{t+1} - q_i \|^2 \Rightarrow (39). \]

Here, we used Lemma 1, \( \mu \)-convexity of \( H \) (15), Cauchy-Schwartz inequality, and the definition of \( \| \nabla H \|_{\text{max}} \). Substituting (39) in (37), we have

\[ \Delta_{\text{global}} \leq \left( \| \nabla J \|_{\text{max}} + \frac{2 L_j \| \nabla H \|_{\text{max}}}{\mu} \right) \| q_{t+1} - \xi_{t+1} \| + \frac{L_j L_H}{\mu} \min_{i \in [t]} \| q_{t+1} - q_i \|^2. \]

(40)

Next, we characterize an upper bound for the projection distance \( \| q_{t+1} - \xi_{t+1} \| \) at every such iteration \( t \notin \Pi \). From (27), we have \( \| \xi_{t+1} - q_{t+1} \| \leq \| \xi_{t+1} - q_i \| \) for every \( i \in [t] \) and \( t \notin \Pi \). Consequently, \( \| \xi_{t+1} - q_{t+1} \| \leq \sqrt{\min_{i \in [t]} \| \xi_{t+1} - q_i \|^2} \). Using (39),

\[ \| q_{t+1} - \xi_{t+1} \| \leq \sqrt{\frac{2 \| \nabla H \|_{\text{max}}}{\mu} \| q_{t+1} - \xi_{t+1} \| + \frac{L_H}{\mu} \min_{i \in [t]} \| q_{t+1} - q_i \|^2}. \]

(41)
Assume, for contradiction, that there is some first-order sequential optimization algorithm with $\mu > (36)$. To show that Algorithm 3 can fail to solve (1) when $\mu$ violates.

Finally, substituting (42) into (40), we obtain

$$\|q_{t+1} - \xi_{t+1}\| \leq \frac{a}{2a} \leq \frac{L_H}{2\|\nabla H\|_{\max}} \min_{i \in [t]} \|q_{t+1} - q_i\|^2.$$

(42)

Finally, substituting (42) into (40), we obtain

$$\Delta_{\text{global}} \leq \kappa \min_{i \in [t]} \|q_{t+1} - q_i\|^2,$$

where $\kappa \leq L_J \left( \frac{L_H \|\nabla J\|_{\max}}{2L_J \|\nabla H\|_{\max}} + \frac{2L_H}{\mu} \right).$

(43)

The upper bound (43) also upper bounds (38), since $\frac{\kappa}{\mu} \geq \frac{2L_H}{\mu} \geq 2$. Therefore, we can guarantee $\Delta_{\text{global}} \leq \eta$, when $T$ is chosen such that for some $t \in [T-1]$, $\min_{i \in [t]} \|q_{t+1} - q_i\|^2 \leq \frac{\eta}{2}$. We complete the proof using Lemma 2 and (36).

3.2.1 Tightness of the worst-case analysis

Proposition 3 shows that Algorithm 3 is worst-case optimal in the user-specified suboptimality threshold $\eta$. We only focus on first-order sequential optimization algorithms that can solve (1) with $\mu$-convex $H$ and guarantee no constraint violation.

Proposition 3. For every first-order sequential optimization algorithm that solves (1) without producing any queries in the infeasible set $H > 0$, $K_J \geq 0$, and $L_H \geq 0$ there is a problem instance of (1) with twice-differentiable, smooth $J \in F_{K_J}$, and $\mu$-convex, smooth $H \in F_{\mu}$ such that the algorithm requires at least $\left\lceil \left( \frac{K_J}{\mu} \right)^{\frac{1}{2}} C' \right\rceil$ queries to compute an $(0, \eta, 0)$-minimum. Here, $C'$ is a positive constant independent of true Lipschitz gradient constant of the objective function $K_J$ and $\eta$.

Proof. We first note that, under the given assumptions, (1) is equivalent to the following optimization problem,

$$\min_j J(x) \quad \text{subject to } x \in \mathcal{Y} \triangleq \{ H \leq 0 \} \cap \mathcal{X}'.$$

(44)

By definition, the set $\mathcal{Y}$ is closed (since $H$ is continuous), convex (since $H$ is convex), and bounded (since $H$ is $\mu$-convex with $\mu > 0$) and $\mathcal{X}'$ is convex and compact. Thus, (44) is similar to (1) with $\mathcal{X}'$ restricted to a priori unknown, convex, and compact set $\mathcal{Y}$. Recall that for every first-order sequential optimization algorithm designed to solve (1) when $\mathcal{Y}$ is known, there is a twice-differentiable, smooth objective function $J$ for which the algorithm takes $\left\lceil \left( \frac{K_J}{\mu} \right)^{\frac{1}{2}} C' \right\rceil$ queries to compute an $(0, \eta, 0)$-minimum for some positive constant $C' > 0$ [29, Thm. 4]. Clearly, the necessary bound must also hold for the case where the set $\mathcal{Y}$ is a priori unknown, and the algorithms query only within the set $\mathcal{Y}$. This completes the proof.

From Theorem 11 and Proposition 3, the sufficient budget $T_{\text{sufficient-} \mu\text{-convex}}$ is necessary and sufficient (up to a constant factor) to address Problem 3 irrespective of the choice of the objective and the constraint function or the feasible initial solution.

3.2.2 Is strong-convexity of $H$ necessary in Problem 3?

Assume, for contradiction, that there is some first-order sequential optimization algorithm $\mathcal{A}$ that solves (1) without any constraint violation and requiring $H$ to be both smooth and not necessarily strongly-convex. By Whitney’s theorem, for every constraint function $H$ and the associated sequence of feasible iterates $\{q_i\}_{i \in [T]}$ generated by $\mathcal{A}$, there exists a constraint function $H' \in F_{L_H}$ such that the first-order oracles of $H$ and $H'$ agree at all iterations $i \in [t-1]$ for some $t \in [T]$, but $H'(q_i) > 0 \geq H(q_i)$. In other words, there always exist a problem instance for which $\mathcal{A}$ will violate the constraint at iteration $t$, specifically the problem instance (1) with $H'$ as the constraint function instead of $H$.

To show that Algorithm 3 can fail to solve (1) when $H$ is just convex, but not strongly-convex, consider the constraint function as the zero function $H \triangleq 0$. In this case, every iterate of Algorithm 3 is $q_1$, the user-provided initial solution guess, since the set $\{ H_1^+(x) \leq 0 \} = \{ q_1 \}$ at every iteration $t \in \mathbb{N}_+$. 
3.3 Tractable implementation of Algorithms 2 and 3

Algorithms 2 and 3 require global optimization of non-convex optimization problems (23), (26), and (27). We now discuss tractable approaches to solve these optimization problems.

### For Algorithm 2

- **minimize** \( L_J(u - \frac{z}{r}) \)
- subject to \( u \in \mathbb{R} \), \( x \in \mathcal{X} \)
- \( \forall i \in [t], \ell(x; q, J) \leq u \)
- \( \forall j \in [t], \ell(x; q, H) \leq \frac{r}{2} \)

### For Algorithm 3

- **minimize** \( L_J(u - \frac{z}{r}) \)
- subject to \( u \in \mathbb{R} \), \( x \in \mathcal{X} \)
- \( \forall i \in [t], \ell(x; q, J) \leq u \)
- \( \forall j \in [t], \ell(x; q, H) \leq \frac{r}{2} \)

#### Definition

Figure 3: Reformulation of (23) and (26). Both of the optimization problems minimizes a concave quadratic function subject to convex constraints from \( \mathcal{X} \), \( t \) linear constraints, and \( f \) quadratic constraints. While the quadratic constraints in (23) are non-convex, the quadratic constraints in (26) are convex.

Figure 4 sketches a reformulation of (23) and (26). The resulting problems are non-convex, quadratically-constrained quadratic programs, when the constraint \( x \in \mathcal{X} \) can be expressed as a collection of linear/second-order cone constraints. We utilize Gurobi, a commercial off-the-shelf solver, to solve such problems. Gurobi can tackle (23) and (26) via spatial branching [9]. The reformulation of (23) also shows that Algorithm 2 simplifies to a minimax space-filling design-based optimization (23) for very large \( L_J \) and \( L_H \).

The optimization problem (27) seeks the projection of a point \( \xi_{t+1} \in \mathcal{X} \) onto the set \( \{H_i^+ \leq 0\} \) at every iteration \( t \in [T] \). From (10), and simple algebraic manipulations, we see that

\[
\{H_i^+ \leq 0\} = \bigcup_{i \in [t]} \text{Ball} \left( q_i - \frac{\nabla H(q_i)}{L_H}, \sqrt{\frac{\|
abla H(q_i)\|^2}{L_H^2} - \frac{2H(q_i)}{L_H}} \right)
\]

(45)

Consequently, we can solve (27) exactly in two steps: 1) compute the projection point of \( \xi_{t+1} \) onto the \( t \) balls separately (available in closed-form), and 2) choose among the \( t \) projection points, the point closest to \( \xi_{t+1} \) via a finite minimum operation. Recall that for any \( z \in \mathcal{X} \), the projected point is \( z \), if \( z \in \text{Ball}(c, r) \), otherwise, the projected point is \( c + \frac{r}{\|z - c\|}(z - c) \).

4 Numerical experiments

We used Python to perform all computations on an Intel i7-4600U CPU with 4 cores, 2.1GHz clock rate and 7.5 GB RAM.

4.1 Benchmarking against existing approaches: Solution quality and scalability

We consider several benchmark problems to compare the performance of Algorithms 2 and 3 with existing approaches to solve (1) — bayesian optimization and local optimization. For bayesian optimization, we considered the constrained expected improvement (CEI) (7) approach as implemented in emukit (20). emukit solves the resulting unconstrained, non-convex, acquisition optimization problem approximately using Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm via random starts. We also considered SLQP, a first-order local optimization algorithm as implemented in Python’s scipy package (30).
We investigate the quality of the computed solutions in terms of their global suboptimality, the compute time, and the

Table 2 lists the eight benchmark problems in the form of (1). Here, we chose the set

and 3 do not dominate each other. The global suboptimality bound \( \Delta_{\text{global}} \) computed by Algorithms 2 and 3 at each iteration upper bounds the true global suboptimality \( J(x^*) - J(x_t^*) \), and demonstrates the anytime property of these algorithms. Empirically, we see that the upper bound \( \Delta_{\text{global}} \) is not severely conservative, and tracks the decrease in the true global suboptimality well. The upper bound \( \Delta_{\text{global}} \) helps Algorithms 2 and 3 terminate early in Problems \( P_2, P_3, P_6 \), and \( P_8 \), while guaranteeing the satisfaction of the desired global suboptimality threshold of \( \eta = 0.1 \). In most of the problems, the trials of the Bayesian optimization approach (cEI) and the iterates of Algorithms 2 and 3 do not dominate each other.

| Problem | Function | \( K_f \) | \( L_f \) |
|---------|----------|-----------|-----------|
| P1      | Br \( J \) | 10        | 2         |
| P2      | MBr \( H \) | 50        | 10        |
| P3      | Br \( c_{\text{bowl}} + \frac{R_{\text{bowl}}}{2} (1 \sqrt{2}, 1 \sqrt{2}) \) | 50        | 10        |
| P4      | MBr \( (5.5, -9) \) | 50        | 10        |
| P5      | Br \( (5.5, -9) \) | 50        | 10        |
| P6      | MBr \( (5.5, -9) \) | 50        | 10        |
| P7      | Br \( (5.5, -9) \) | 50        | 10        |
| P8      | MBr \( (5.5, -9) \) | 50        | 10        |

Table 2: Benchmark problems defined using two-dimensional functions with \( \mathcal{X} = [-10, 10]^2 \). The true Lipschitz constant for the gradients \( K_f \), when unknown, are computed via gridding. See [14 Sec. B] for the definition of the Branin function. We use strong-convexity constant \( \mu = 0.5 \) for \( \text{Bowl} \) function, which is smaller than its true strong-convexity constant of one.

We investigate the quality of the computed solutions in terms of their global suboptimality, the compute time, and the number of infeasible queries. We also discuss the near-infeasibility certificates computed by Algorithm 3 and its ability to deal with moderately-dimensioned problems.

4.1.1 Solution quality

Table 2 lists the eight benchmark problems in the form of (1). Here, we chose the set \( \mathcal{X} = [-10, 10]^2 \), the thresholds \( \eta = 0.1 \) and \( \delta = 10^{-5} \), and a budget of \( T = 400 \).

For the Bayesian optimization, we considered 5 independent trials to account for the stochastic behavior of the emukit’s implementation of cEI. We used grid search followed by a “polishing step” using local optimization SLSQP to approximate the true global minimum of (1) \( J(x^*) \) by \( J(x_t^*) \). For grid search, we used a step size of 0.05, which resulted in 40,000 oracle queries (excluding the queries in the polishing step).

Figure 3 summarizes the results. We see that Algorithm 2 computes the (5-relaxed) global optima within the budget for every problem in Table 2 while Algorithm 3 computes the global optima without any constraint violation whenever the unknown constraint function \( H \) is strongly-convex (Problems \( P_7 \) and \( P_8 \)). Both of the proposed algorithms are significantly faster than Bayesian optimization (cEI) in most of the trials. In addition, Bayesian optimization currently lacks the guarantee constraint-violation-free optimization when \( H \) is strongly-convex, and produces infeasible queries in contrast to Algorithm 3. As expected, the local optimization method SLSQP can return suboptimal solutions (Problems \( P_1, P_3 \), and \( P_5 \)), but converges significantly faster than Algorithms 2 and 3 and Bayesian optimization. Unlike the proposed algorithms, SLSQP can converge to an infeasible solution (Problems \( P_7, P_5, \) and \( P_5 \)).
Constrained, Global Optimization of Functions with Lipschitz Continuous Gradients

Figure 4: Results of the benchmark problems (Table 2) for a budget $T = 400$. Algorithm 2 computes a $(0.1, 10^{-5})$-minimum for every problem, irrespective of the feasibility of the initial solution guess. For Problems $P_7$ and $P_8$ with a priori unknown, strongly-convex constraint $H \leq 0$, Algorithm 3 computes a $(0, 0)$-minimum without any constraint violation. Both of the algorithms return an upper bound $\Delta_{\text{global}}$ on the true global suboptimality of the computed solution, $J(x^\ast) - J(x^g)$. They outperform Bayesian optimization (cEI) in computation time, and recover a global minimum unlike SLSQP, a local optimization method.

4.1.2 (Near-)Infeasibility certificates

Algorithm 2 can produce infeasibility or near-infeasibility certificates for infeasible instances of (1). To illustrate the utility of such a certificate, consider the following infeasible optimization problem in $x$,

$$\min_{x} J(x) = MBr(x) \quad \text{subject to} \quad H(x) = Br(x) \leq 0.$$  (46)

The infeasibility proof of (46) follows from the fact that the global minimum value of $Br(x)$ is strictly positive [14, Sec. B.3], which implies that $\{Br(x) \leq 0\} = \emptyset$. We chose $X = [-10, 10]^2$ and a budget of $T = 400$. Using the modification of Algorithm 2 given in (33), we found $\gamma = 0.32$ after exhausting the budget in 58.05 seconds (< 1 minute). In other words, Algorithm 2 proves that

$$\min_{x} J(x) = MBr(x) \quad \text{subject to} \quad H(x) = Br(x) \leq -0.32,$$  (47)

is infeasible (Definition 3). Without the modification, the near-infeasibility certificate returned by Algorithm 2 is much higher $\gamma = 256.98$ after exhausting the budget in 97.33 seconds (< 2 minutes).

4.1.3 Scalability

For scalability evaluation for $d \in \{2, 3, 4, 5\}$, we considered the optimization problem (1) with $J$ as the $d$-dimensional Rosenbrock’s function and constraint $H = \text{InvBowl}$. Recall that the global minimum value $x^\ast$ of $J$ over $X = [-10, 10]^d$ is $1_d$, a $d$-dimensional vector of ones [14, Sec. B.6]. We set constraint $H = \text{InvBowl}$ with $c_{\text{Bowl}}$ as the mid point of line joining the global minimum and the initial point $q_1 = [-10, -10, -10, …] \in \mathbb{R}^d$ (one of the vertices of $X$). We chose $R_{\text{Bowl}} = 0.4\|x^\ast - q_1\|$ to ensure that the selected initial point $q_1$ remains feasible. We chose $L_J = 60$ and $L_H = 2$.

Figure 6 shows that Algorithm 2 can be applied to (1) with moderate values of $d$ as well. While the actual suboptimality of the optimization problem remains low, we found that the suboptimality bounds become loose as $d$ increases, potentially due to $\eta^{-d/2}$ dependence on the sufficient budget (Theorem 1D). In addition, the computational time of Algorithm 2 increases with $d$, potentially due to the reliance on mixed-integer optimization to solve (23).

4.2 Training a neural network with constraints

Next, we apply Algorithm 2 to an instance of (1) arising from policy optimization in machine learning. Specifically, we consider the policy optimization for the classical mountain car problem [3], where we train a policy neural network.
Figure 5: Global suboptimality of iterates returned by Algorithms 2 and 3 and Bayesian optimization (cEI). Algorithms 2 and 3 provide a global suboptimality upper bound $\Delta_{\text{global}}$ at every iteration. In Problems $P_2$, $P_4$, $P_6$, and $P_8$, $\Delta_{\text{global}}$ enables early termination, resulting in significant gains in computation time.
NN(θ) with five network parameters. We seek a policy that drives the car to reach the top of the mountain within a predetermined number of steps.

**Policy neural network:** We consider a network with two input nodes and one output nodes using tanh activation function. The network is defined as follows,

\begin{align}
y_1 &= \tanh(\theta_1 \times \text{car\_position}) \\
y_2 &= \tanh(\theta_2 \times \text{car\_velocity}) \\
u &= \tanh(\theta_3 y_1 + \theta_4 y_2 + \theta_5)
\end{align}

(48a, 48b, 48c)

Here, \(\theta_i\) are the network parameters or the decision variables. We fixed the domain of network parameters \(\Theta = \{ z : \theta_{\text{max}} \leq z \leq \theta_{\text{max}} \} \subset \mathbb{R}^5 \) with \(\theta_{\text{max}} = 5 \times [1.2, 0.67, 1.1, 1, 1]\). Here, we have normalized states (position and velocity of the car) by their bounds.

**Physics-driven constraints:** In general, policy optimization is a hard problem [28], since the mapping from the policy parameters to the reward is highly non-convex, and the large number of policy parameters prevents tractable enumeration. A natural way to reduce the search space is to enforce additional constraints on the problem. Consider the following policy optimization problem,

\[\max_{\theta \in \Theta \subset \mathbb{R}^5} J = \text{CumulativeRewardOverAnEpisode}(\text{NN}(\theta)), \quad \text{s.t.} \quad H = \text{TotalEnergyAtEndOfEpisode}(\text{NN}(\theta)) \geq \text{GoalPotentialEnergy.}\]

The motivation for imposing constraints on the total energy arise from the observation that successful policies that drive the car to the top of the mountain should also inject sufficient energy into the car. Here, we compute the total energy of the system and the goal potential energy at the end of the episode as follows with \(g = 9.8\),

\[
\text{TotalEnergyAtEndOfEpisode}(\text{NN}(\theta)) = \text{Potential energy} + \text{Kinetic energy} = g \times \text{Height(terminal\_car\_position)} + \frac{\text{terminal\_car\_velocity}^2}{2}
\]

\[
\text{GoalPotentialEnergy} = g \times \text{Height(goal\_position\_x)}
\]

We can safely ignore the mass of the car since it appears on both sides of the constraint. We declare that the task is completed successfully, when the cumulative reward is above 90 [3].

Note that \(J\) and \(H\) are smooth functions of the policy parameters \(\theta\), since we have used \(\tanh\) as the activation function in (48). We compute the gradients \(\nabla J\) and \(\nabla H\) via finite differences (step size of 0.01) and choose sufficiently large \(L_J = L_H = 100\). We chose a budget of \(T = 10\), which translates to 110 episodes for finite difference-based gradient computation.

We found that Algorithm 2 computed a policy neural network completes the task successfully. On the other hand, when the energy constraints were not imposed, we did not meet the minimum reward threshold for success, possibly due to the low number of episodes.
5 Conclusion

This paper introduces two novel algorithms for constrained global optimization of a priori unknown functions with Lipschitz continuous gradients. The proposed approaches are inspired by the existing literature in covering method to global optimization problems. They accommodate finite budget of oracle calls and terminate with non-trivial global suboptimality guarantees. The first approach accommodates infeasible start and returns near-global minimum or a (near-)infeasibility certificate. The second approach guarantees feasible iterates when the unknown constraint function is strongly-convex and the initial solution guess is feasible. We also characterize the necessary and sufficient budget of oracle calls required to satisfy user-specified tolerances for a large class of optimization problems. Empirical studies show the efficacy of these approaches.

A Adversarial instance of $H$ in Section 2.1

We construct a resistive oracle for the constraint functions to meet the requirements specified in Section 2.1 and it suffices to consider a single-constraint case $M = 1$. Resistive oracles for sequential optimization algorithms do not commit to a specific $H$, but adapt based on the queries. Analyzing the algorithm’s performance under such oracles reveals its worst-case performance. See [18] for more details.

Desirable properties of the resistive oracle for $H$: Given $T \in \mathbb{N}_+$, $L_H > 0$, and $T$ oracle queries arising from any sequential optimization algorithm, we can construct a first-order oracle for some $H \in \mathcal{F}_{L_H}$ such that:

1. all of the $T$ oracle calls returns $H > 0$ and $\nabla H = 0$, i.e., all of the query points requested by a sequential optimization algorithm are infeasible for (1), and
2. there exists $y \in \mathcal{X}$ distinct from the $T$ query points such that $y$ is feasible for (1).

In other words, given $T$ and $L_H$, the constructed oracle responds to the queries of any sequential optimization algorithm such that the algorithm can “discover” the feasibility of the constrained optimization problem (1), only at the $(T + 1)^{th}$ query. Since every algorithm is bound by the budget of the oracle calls, it is forced to declare infeasibility based on the infeasible $T$ queries.

Construction of the resistive oracle for $H \in \mathcal{F}_{L_H}$: Let $Q_T \triangleq \{q_i : i \in [T]\}$ be the set of query points corresponding to the first $T$ oracle calls from the sequential algorithm under study. We define $y \in \arg \sup_{x \in \mathcal{X}} \min_{i \in [T]} \|x - q_i\| \in \mathcal{X}$, a point in $\mathcal{X}$ that is the furthest away from $Q_T$. By Whitney’s extension theorem [31], there is always a function $h : \mathcal{X} \to \mathbb{R}$ with Lipschitz continuous gradient, such that $h(q_i) > 0$ and $\nabla h(q_i) = 0$ for $i \in [T]$, and $h(y) \leq 0$. While the constructed $h$ need not lie in $\mathcal{F}_{L_H}$ as desired, we can always construct the desired $H \in \mathcal{F}_{L_H}$ via $H = \alpha h$ for some appropriate scaling $\alpha > 0$. This completes the construction.

B Illustrative example on page 8

For the first example, we study the following non-convex optimization problem,

$$\begin{align*}
\text{minimize} & \quad J(x) = \frac{\sin(x)}{2x} - 0.02x \\
\text{subject to} & \quad x \in \mathcal{X} = [-10, 10], \quad H(x) = \frac{(x-6)^2(x+6)^2-900}{4000} \leq 0
\end{align*}$$

(49)

with Lipschitz gradient constants as $L_J \in \{0.2, 1\}$ and $L_H = 0.2$. We choose suboptimality threshold $\eta = 0.01$, and relaxation threshold $\delta = 10^{-8}$.

For the second example, we study the following non-convex optimization problem with strongly-convex constraint function $H$,

$$\begin{align*}
\text{minimize} & \quad J(x) = \frac{\sin(x)}{2x} - 0.02x \\
\text{subject to} & \quad x \in \mathcal{X} = [-10, 10], \quad H(x) = \frac{(x-1)^2-T^2}{100} \leq 0
\end{align*}$$

(50)

with Lipschitz gradient constants as $L_J = 0.2$ and $L_H = 1.2$, and convexity constant $\mu = 0.01$. We choose suboptimality threshold $\eta = 0.01$.

References

[1] R. Blanquero and E. Carrizosa, On covering methods for D.C. optimization, J. Global Optim., 18 (2000).
[2] S. Boyd and L. Vandenberghe, Convex optimization, Cambridge Univ. Press, 2004.
[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym, 2016, https://arxiv.org/abs/arXiv:1606.01540.

[4] N. de Freitas, A. J. Smola, and M. Zoghi, Exponential regret bounds for gaussian process bandits with deterministic observations, in Proc. Int. Conf. Mach. Learn., 2012, p. 955–962.

[5] D. Eriksson and M. Poloczek, Scalable constrained bayesian optimization, arXiv preprint arXiv:2002.08526, (2020).

[6] D. Fortin and I. Tseveendorj, Piecewise Convex Maximization Problems: Piece Adding Technique, J. Optim. Theory Appl., 148 (2011), pp. 471–487.

[7] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, Bayesian optimization with inequality constraints, in Proc. Int. Conf. Mach. Learn., 2014, pp. 937–945.

[8] M. A. Gelbart, J. Snoek, and R. P. Adams, Bayesian optimization with unknown constraints, in Conf. Uncertain. Artificial Intell., 2014, pp. 250–259.

[9] Gurobi Optimization LLC, Gurobi optimizer reference manual. http://www.gurobi.com.

[10] R. Horst, P. M. Pardalos, and N. Van Thoai, Introduction to global optimization, Springer Science & Business Media, 2000.

[11] V. Ivanov, Optimal algorithms of minimization of certain classes of functions, Cybernetics, 8 (1972), pp. 620–634.

[12] M. Jaggi, Revisiting Frank-Wolfe: Projection-free sparse convex optimization, in Proc. Int. Conf. Mach. Learn., vol. 28, Atlanta, Georgia, USA, 17–19 Jun 2013, pp. 427–435.

[13] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, Lipschitzian optimization without the lipschitz constant, Journal of optimization Theory and Applications, 79 (1993), pp. 157–181.

[14] M. J. Kochenderfer and T. A. Wheeler, Algorithms for optimization, MIT Press, 2019.

[15] C. Malherbe and N. Vayatis, Global optimization of lipschitz functions, in Proc. Int. Conf. Mach. Learn., 2017, pp. 2314–2323.

[16] J. Mockus, Bayesian approach to global optimization: theory and applications, vol. 37, Springer Science & Business Media, 2012.

[17] R. Munos, Optimistic optimization of a deterministic function without the knowledge of its smoothness, in Adv. Neural Info. Process. Syst., 2011, pp. 783–791.

[18] Y. Nesterov, Lectures on Convex Optimization, Springer International Publishing, 2018.

[19] J. Nocedal and S. J. Wright, Numerical optimization, Springer series in operations research, Springer, New York, 2nd ed., 2006.

[20] A. Paleyes, M. Pullin, M. Mahsereci, N. Lawrence, and J. González, Emulation of physical processes with emukit, in Second Workshop Mach. Learn. Phys. Sci., Adv. Neural Info. Process. Syst., 2019.

[21] P. M. Pardalos, Q. P. Zheng, and A. Arulselvan, Deterministic global optimization, Encycl. Oper. Res. Management Sci., (2010).

[22] S. Piyavskii, An algorithm for finding the absolute extremum of a function, Comput. Math. Math. Phys., 12 (1972), pp. 57–67.

[23] L. Pronzato, Minimax and maximin space-filling designs: some properties and methods for construction, Journal de la Société Française de Statistique, 158 (2017), pp. 7–36.

[24] C. E. Rasmussen and C. Williams, Gaussian processes for machine learning, vol. 1, MIT press, 39 (2006), pp. 40–43.

[25] B. O. Shubert, A sequential method seeking the global maximum of a function, SIAM J. Numer. Anal., 9 (1972), pp. 379–388.

[26] J. Snoek, H. Larochelle, and R. P. Adams, Practical bayesian optimization of machine learning algorithms, in Adv. Neural Info. Process. Syst., 2012, pp. 2951–2959.

[27] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, Information-theoretic regret bounds for gaussian process optimization in the bandit setting, IEEE Trans. Inform. Theory, 58 (2012), pp. 3250–3265.

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

[29] S. A. Vavasis, Complexity issues in global optimization: A survey, in Handbook of global optimization, Springer, 1995, pp. 27–41.
[30] P. Virtanen et al., *Scipy 1.0: Fundamental algorithms for scientific computing in python*, Nature Methods, 17 (2020), pp. 261–272.

[31] H. Whitney, *Analytic extensions of differentiable functions defined in closed sets*, Trans. Amer. Math. Soc., 36 (1934), pp. 63–89.

[32] J. Wu, M. Poloczek, A. G. Wilson, and P. Frazier, *Bayesian optimization with gradients*, in Adv. Neural Info. Process. Syst., Curran Associates, Inc., 2017, pp. 5267–5278.

[33] D. Yudin and A. Nemirovski, *Problem complexity and method efficiency in optimization*, Wiley-Interscience, 1983.