Information Preserving Line Search via Bayesian Optimization

Robin Labryga[1], Tomislav Prusina[1], and Sören Laue[1]

University of Hamburg, Germany
{robin.labryga,tomislav.prusina,soeren.laue}@uni-hamburg.de

Abstract. Line search is a fundamental part of iterative optimization methods for unconstrained and bound-constrained optimization problems to determine suitable step lengths that provide sufficient improvement in each iteration. Traditional line search methods are based on iterative interval refinement, where valuable information about function value and gradient is discarded in each iteration. We propose a line search method via Bayesian optimization, preserving and utilizing otherwise discarded information to improve step-length choices. Our approach is guaranteed to converge and shows superior performance compared to state-of-the-art methods based on empirical tests on the challenging unconstrained and bound-constrained optimization problems from the CUTEst test set.

Keywords: Nonlinear Optimization · Line Search · Regression · Bayesian Optimization · Gaussian Process.

1 Introduction

Line search is an essential part of iterative optimization algorithms [11,26,32], determining step lengths that lead to sufficient improvement in each iteration. Traditional methods rely on iterative interval refinement, where an interval containing a potential solution is progressively narrowed by evaluating function values and gradients. While this process systematically reduces the search space, it discards previously obtained information after each iteration. This can result in missing better steps, especially when the refined interval no longer includes a step that offers greater improvement.

This paper introduces a novel approach to inexact line search that leverages Bayesian optimization. Unlike traditional methods, which discard information after each refinement, our approach preserves and utilizes all observed data to make more informed decisions about step lengths. By approximating the objective function with a Gaussian process surrogate model, our method effectively identifies steps that can provide better improvements. It is well-suited for unconstrained or bound-constrained, continuously differentiable black-box objective functions that are bounded from below. Furthermore, the approach is especially advantageous when function and gradient evaluations are expensive.

1.1 Related Work

Numerous line search methods have been developed to ensure convergence and improve efficiency. Some widely used methods include backtracking line search [26], which iteratively reduces the step size until the sufficient decrease condition is met. Hager's line search [13] combines bracketing, bisection, and Newton iterations based on cubic interpolation. The Moré-Thuente line search [25] employs interpolation and an auxiliary function to satisfy the strong Wolfe conditions. The filter line search method by Wächter and Biegler [38] is used in interior-point methods to guide the search by considering both objective function reduction and constraint violations. These methods are widely adopted due to their simplicity and ability to satisfy Wolfe or strong Wolfe conditions [26,39,40]. However, they rely on interval refinement, which inherently discards valuable information from intermediate evaluations.

The Barzilai-Borwein line search [2] is a non-monotone step size selection method that estimates the step length using differences in gradients and iterates, inspired by quasi-Newton updates. Unlike traditional line search methods that enforce Wolfe conditions, it adaptively selects step sizes without backtracking, making it particularly effective for quadratic and large-scale unconstrained problems. However, it often requires stabilization to maintain robustness for highly nonlinear objectives.

More recently, Prusina and Laue [28] introduced a surrogate-based line search using cubic interpolation, which retains more information than classical methods. However, this approach remains limited to predefined interpolation models and lacks theoretical convergence guarantees. Similarly, Papageorgiou et al. [27] proposed a surrogate-based line search method tailored for smooth and derivative-free optimization problems, further expanding the scope of surrogate-based approaches.

Bayesian optimization has emerged as a powerful tool in global optimization, particularly for expensive black-box objective functions [9,10]. It has been successfully applied in various domains, including hyperparameter tuning for machine learning models [31], engineering system design [7], and experimental design selection [3]. While its use in global optimization is well-documented (e.g., [23,24]), its application to line search processes has received limited attention. Mahsereci and Henning [22] proposed a Bayesian optimization-based line search for stochastic optimization, highlighting the potential of Bayesian methods in this context.

Building on this foundation, our approach leverages Bayesian optimization in the deterministic optimization setting, to systematically model function values and gradients. Unlike classical line search methods that discard intermediate evaluations, our method preserves and utilizes all available information, addressing a critical gap in the literature.

1.2 Contributions

This paper makes the following contributions:

- **Novel Line Search Framework**: We propose a Bayesian optimization-based line search method that preserves and utilizes all available data to make more informed step-length decisions.
- **Convergence Proof**: We provide a theoretical proof of convergence for our method, ensuring it can reliably identify sufficient improvements in the optimization process.
- **Extensive Empirical Evaluation**: Our method is integrated into the GENO solver [19,21] and benchmarked against state-of-the-art methods, demonstrating superior performance on challenging unconstrained and bound-constrained problems from the CUTEst test set [12].

The code for our Bayesian optimization-based line search method is available on GitHub[1].

This paper is organized as follows: Section 2 provides an overview of line search, Bayesian optimization, and Gaussian processes. Section 3 details our proposed approach, including its theoretical foundation, convergence proof, and implementation. Section 4 presents empirical evaluations using the GENO solver [19] on benchmark problems from the CUTEst test set [12]. Our results demonstrate superior performance compared to state-of-the-art methods.

2 Background

This section provides an overview of the foundational concepts underlying our approach, including line search, Bayesian optimization, and Gaussian processes. These concepts form the basis for the proposed approach of utilizing Bayesian optimization for line search.

2.1 Line Search

Consider a continuously differentiable objective function $f\colon \mathbb{R}^n \to \mathbb{R}$ to be minimized. In iterative optimization, line search starts from a point $x \in \mathbb{R}^n$ and moves in a descent direction p with $p^\top \nabla f(x) < 0$. The aim is to find a step length α along p that sufficiently improves upon f. The univariate line search objective ϕ and its derivative ϕ' along p are defined as

$$\phi(\alpha) \coloneqq f(x + \alpha \cdot p), \quad \phi'(\alpha) = p^\top \nabla f(x + \alpha \cdot p), \quad \alpha > 0 \tag{1}$$

Bounds on the step α, such as a maximal step α_{max}, may also be imposed.

Different line search methods have varying requirements for the chosen step. The sufficient decrease condition (2), also known as the Armijo rule [1], requires that the function value at α is below a flattened tangent from the starting point, ensuring sufficient decrease:

$$\phi(\alpha) \leq \phi(0) + \mu \cdot \phi'(0) \cdot \alpha, \quad \mu \in (0,1) \tag{2}$$

[1] https://github.com/RobinLabryga/bayesian-geno/tree/bayesian-ls-paper

The curvature condition (3) ensures that the negative gradient at α is not too large compared to the starting point, preventing the step from being too short:

$$-\phi'(\alpha) \leq \eta \cdot -\phi'(0), \quad \eta \in (0,1) \tag{3}$$

However, the curvature condition does not guarantee that the gradient at α is closer to zero than at the starting point. The modified curvature condition (4) addresses this by requiring the absolute value of the gradient at α to be sufficiently smaller than at the starting point:

$$|\phi'(\alpha)| \leq \eta \cdot |\phi'(0)|, \quad \eta \in (0,1) \tag{4}$$

The combination of the sufficient decrease condition (2) and the curvature condition (3) is known as the Wolfe conditions, while the combination of the sufficient decrease condition (2) with the modified curvature condition (4) is known as the strong Wolfe conditions [26,39,40]. Typically, the parameters η and μ for the (strong) Wolfe conditions are chosen such that $\mu < \eta$ (e.g., $\mu = 10^{-4}, \eta = 0.9$) [26].

Optimizers using a line search method that finds a step satisfying the strong Wolfe conditions are convergent if the search direction p is a descent direction. Gradient methods and quasi-Newton methods satisfy this criterion [5,26]. Additionally, the strong Wolfe conditions ensure a positive definite quasi-Newton update to the Hessian approximation is possible [26].

To guarantee that a step satisfying the strong Wolfe conditions exists within an interval, the auxiliary function ψ [25] is often used:

$$\psi(\alpha) \coloneqq \phi(\alpha) - (\phi(0) + \mu \cdot \phi'(0) \cdot \alpha), \qquad \psi'(\alpha) = \phi'(\alpha) - \mu \cdot \phi'(0), \tag{5}$$

where μ is the parameter from the sufficient decrease condition (2). We have the following theorem:

Theorem 1 (Moré-Thuente 2.1 [25]). *Let I be a closed interval with endpoints α_l and α_u. If the endpoints satisfy*

$$\psi(\alpha_l) \leq \psi(\alpha_u), \quad \psi(\alpha_l) \leq 0, \quad \psi'(\alpha_l)(\alpha_u - \alpha_l) < 0,$$

then there is an $\alpha^\star$ in I with $\psi(\alpha^\star) \leq \psi(\alpha_l)$ and $\psi'(\alpha^\star) = 0$. $\alpha^\star$ satisfies the strong Wolfe conditions.

From this, we obtain the following corollary:

Corollary 1. *Let $I = [\alpha_l, \alpha_u]$ be a closed interval with $\alpha_l < \alpha_u$. If α_l satisfies*

$$\psi(\alpha_l) \leq 0, \quad \psi'(\alpha_l) < 0,$$

and for α_u we have $\psi(\alpha_u) \geq \psi(\alpha_l)$ or $\psi'(\alpha_u) \geq 0$ then there is an $\alpha^\star$ in I with $\psi(\alpha^\star) \leq \psi(\alpha_l)$ and $\psi'(\alpha^\star) = 0$. $\alpha^\star$ satisfies the strong Wolfe conditions.

Proof. By Theorem 1, if $\psi(\alpha_u) \geq \psi(\alpha_l)$, an $\alpha^\star$ exists in I with $\psi(\alpha^\star) \leq \psi(\alpha_l)$ and $\psi'(\alpha^\star) = 0$. If $\psi(\alpha_u) < \psi(\alpha_l)$ and $\psi'(\alpha_u) \geq 0$, we rename α_l and α_u to complete the argument similarly. Thus, $\alpha^\star$ satisfies the strong Wolfe conditions. $\square$

2.2 Bayesian Optimization

We utilize Bayesian optimization to minimize the line search objective and determine a suitable step length. By approximating the objective function with a Bayesian surrogate, we iteratively refine our model using all information from previously queried points.

Surrogate methods approximate an expensive-to-evaluate objective function by conditioning a surrogate model on known function evaluations. The surrogate model is then used to define an acquisition function that selects the next evaluation point. Since the surrogate is cheaper to evaluate than the objective function, this approach allows for more evaluations when optimizing the acquisition function, ultimately reducing the number of expensive objective function evaluations [3,9,10]. An algorithmic overview of Bayesian optimization is presented in Algorithm 1.

Algorithm 1 Bayesian Optimization

```
1: for $t \in [1..T]$ do
2:     Condition surrogate model $\mathcal{M}$ on $\mathcal{D}_{1:t-1}$
3:     $x_t \leftarrow \text{argmax}_{x \in A} u(x|\mathcal{M})$
4:     $\mathcal{D}_{1:t} \leftarrow \mathcal{D}_{1:t-1} \cup \{x_t\}$
5: end for
6: return $\text{argmax}_{x \in \mathcal{D}} \phi(x)$ or $\text{argmax}_{x \in A} \mathcal{M}(x)$
```

Bayesian optimization is typically used for multivariate optimization, but in the context of line search, we focus on the univariate case. Given an objective function $\phi \colon \mathbb{R} \to \mathbb{R}$, our goal is to minimize $\phi(\alpha)$ over a feasible set $A \subset \mathbb{R}$. The Bayesian optimization framework iteratively selects query points to evaluate, refining the surrogate model until a satisfactory solution is found.

Each iteration of Bayesian optimization incurs computational overhead due to updating the surrogate model and optimizing the acquisition function. Consequently, Bayesian optimization is most effective when function evaluations are expensive, and only a limited number of evaluations are feasible [9,10].

Bayesian optimization is known to converge under mild assumptions on the surrogate model and the acquisition function. Specifically, as the algorithm produces a dense sequence of observations, the surrogate model increasingly refines its approximation of the objective function, leading to improved estimates of the global minimum [23,24,35].

Common surrogate models in Bayesian optimization include Gaussian processes [17,35] and Wiener processes [18,34]. In our approach, we use Gaussian processes as the surrogate model, which we discuss in the next section.

2.3 Gaussian Process

A Gaussian process is a collection of random variables where any finite subset follows a joint Gaussian distribution. It is fully specified by a mean function μ

and a covariance function k, which define its properties and structure. Formally, a Gaussian process can be written as:

$$\begin{aligned} f(x) &\sim GP(\mu(x), k(p,q)) \\ \mu(x) &= \mathbb{E}[f(x)] \\ k(p,q) &= \mathbb{E}[(f(p)-\mu(p))(f(q)-\mu(q))] \end{aligned}$$

Before any observations are made, a Gaussian process prior is chosen to reflect prior beliefs about the function's behavior. When conditioned on observed function values and gradients, the prior is updated to yield a Gaussian process posterior, that incorporates all available data to refine the function approximation.

Given n observations at points $X = \{x_1, \dots, x_n\}$, with corresponding function values $y = \{y_1, \dots, y_n\}$ and (optionally) gradients $g = \{\nabla f(x_1), \dots, \nabla f(x_n)\}$, the predictive mean $\overline{\mu}$ and covariance $\overline{\text{cov}}$ of the Gaussian process posterior at a new query point x are given by:

$$\begin{aligned} \overline{\mu}(x) &= \mu(x) + K(x,X)\left(K(X,X)+\sigma^2\mathbb{I}\right)^{-1} Y \\ \overline{cov}(x) &= K(x,x) - K(x,X)\left(K(X,X)+\sigma^2\mathbb{I}\right)^{-1} K(X,x) \end{aligned}$$

where $K(X,X)$ is the Gram matrix of the covariance function, and Y represents the observed function values. If gradient information is available, the Gaussian process can be conditioned on both function values and gradients to enhance predictive accuracy.

The covariance function, also known as the kernel, plays a crucial role in determining the behavior of the Gaussian process. Different kernels model different assumptions about the smoothness and variability of the approximated function. In our approach, we use the Matérn kernel [29] with $\nu = \frac{5}{2}$, which provides a balance between smoothness and adaptability:

$$k_{\nu=5/2}(p,q) = \left(1 + \frac{\sqrt{5}|p-q|}{l} + \frac{5|p-q|^2}{3l^2}\right) e^{-\frac{\sqrt{5}|p-q|}{l}}.$$

This kernel is particularly effective for modeling functions with finite differentiability, making it well-suited for line search applications. Gaussian processes with the Matérn kernel provide accurate uncertainty quantification [6], which is essential for making informed decisions in Bayesian optimization.

Differentiability of Gaussian Processes Since we want to optimize acquisition functions defined based on Gaussian processes, having the derivative of Gaussian processes is invaluable. As differentiation is a linear operator, the derivative of a Gaussian process is another Gaussian process that we can compute [20,29].

The derivative of the predictive mean and the derivative of the predictive covariance can be calculated via

$$\begin{aligned}\frac{\partial}{\partial x}\overline{\mu}(x) &= \frac{\partial}{\partial x}\mu(x) + \frac{\partial}{\partial x}K(x,X)\left(K(X,X)+\sigma^2\mathbb{I}\right)^{-1}Y\\ \frac{\partial}{\partial x}\overline{cov}(x) &= \frac{\partial}{\partial x}K(x,x) - 2\cdot\frac{\partial}{\partial x}K(x,X)\left(K(X,X)+\sigma^2\mathbb{I}\right)^{-1}K(x,X)^{\top}\end{aligned}$$

2.4 Acquisition Functions

In Bayesian optimization, an acquisition function determines the next point to evaluate by balancing exploration (searching in uncertain regions) and exploitation (focusing on regions likely to yield optimal values). This process can also be viewed as minimizing a risk function [23]. The choice of acquisition function significantly affects the efficiency of the optimization process.

Examples of acquisition functions include Probability of Improvement (PI) [18], Expected Improvement (EI) [16], Lower Confidence Bound (LCB) [4], knowledge gradient (KG) [8], derivative-enabled Expected Improvement ($d-EI$) [41], and derivative-enabled knowledge gradient ($d-KG$) [41]. Variants of these acquisition functions, such as $GP-LCB$ [33], also exist.

Most of these acquisition functions are designed for derivative-free scenarios, with $d-EI$ and $d-KG$ being the exceptions that explicitly incorporate gradient information. Since acquisition functions depend on the mean and variance of the surrogate Gaussian process, gradients implicitly affect all acquisition functions if gradient information is used to condition the Gaussian process posterior.

The variance is the diagonal of the covariance, i.e., $\mathbb{V}(x) = \text{diag}(\overline{cov}(x))$, and the standard deviation is the square root of the variance, i.e., $\mathbb{S}(x) = \sqrt{\mathbb{V}(x)}$. Then, the Lower Confidence Bound (LCB) acquisition function [4] and its derivative are defined as follows:

$$\begin{aligned}LCB(x) &= -\mu(x) + \kappa\mathbb{S}(x)\\ \frac{\partial}{\partial x}LCB(x) &= -\frac{\partial}{\partial x}\mu(x) + \kappa\frac{\partial}{\partial x}\mathbb{S}(x)\end{aligned}$$

where $\kappa \geq 0$ is a hyperparameter. A larger value of κ increases exploration by prioritizing points with high uncertainty, while a smaller κ focuses more on exploitation by selecting points with lower predicted function values. The LCB acquisition function is particularly useful when minimizing the function with confidence bounds. In our approach, we utilize the LCB acquisition function.

3 Algorithm

We apply Bayesian optimization to the line search objective $\phi(\alpha)$ (Eq. (1)) to find a step size that satisfies the strong Wolfe conditions. Unlike traditional line search methods that discard intermediate evaluations, our approach retains and utilizes all function and gradient information observed during the search. This is achieved by using a Gaussian process model with the Matérn covariance function and selecting step sizes via the Lower Confidence Bound (LCB) acquisition function. An illustrative comparison between the Moré-Thuente line search and Bayesian optimization-based line search is shown in Fig. 1.

While Bayesian optimization offers a more global approach to finding optimal step sizes, it may suffer from slow convergence if applied naively. Additionally, Bayesian optimization requires an interval with finite endpoints, whereas traditional line searches often operate over unbounded intervals (e.g., $[0, \infty)$). We address these challenges by dynamically refining the search interval and integrating Bayesian optimization with an interval update mechanism.

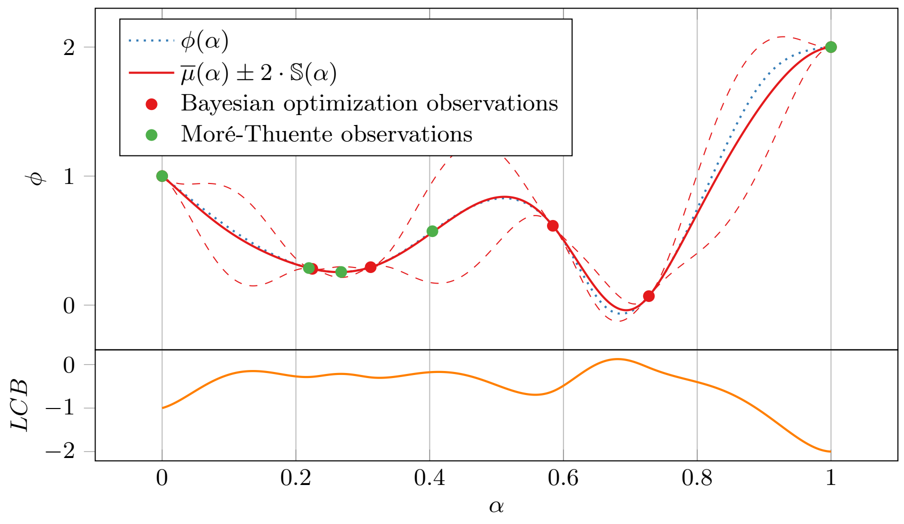


Fig. 1. An exemplary line search comparing the Moré-Thuente line search with Bayesian optimization. The Moré-Thuente line search fails to find the global optimizer, while Bayesian optimization is on track to locate it. The orange line represents the Lower Confidence Bound (LCB) acquisition function, defined based on the Gaussian process conditioned on all prior observations. We observe that the next observation by Bayesian optimization, according to the acquisition function, is very close to the global optimizer.

3.1 Interval Selection

To apply Bayesian optimization effectively, we first determine an initial bounded interval that contains a step satisfying the strong Wolfe conditions. We initialize the search interval as $[\alpha_l = 0, \alpha_u = \alpha_0]$, where $\alpha_0 = \min(1, \alpha_{\max})$. If this interval does not contain a valid step, we expand it iteratively by updating the interval as follows:

$$[\alpha_l, \alpha_u] \leftarrow [\alpha_u, \min(c \cdot \alpha_u, \alpha_{\max})], \quad c \in \mathbb{R}_{>1}$$

If $\psi(\alpha_u) \geq \psi(\alpha_l)$ or $\psi'(\alpha_u) \geq 0$, we can guarantee the presence of a step satisfying the strong Wolfe conditions within the interval according to Corollary 1, since $\alpha_l < \alpha_u$, $\psi(\alpha_l) \leq 0$, and $\psi'(\alpha_l) < 0$. Note that each interval update implies an improvement upon ϕ, as $\psi(\alpha_u) < \psi(\alpha_l)$ and $\alpha_l < \alpha_u$ implies $\phi(\alpha_u) < \phi(\alpha_l)$.

3.2 Bayesian Optimization for Step Selection

Once we have an interval $[\alpha_l, \alpha_u]$ that either contains a valid step or has reached the maximum step size, i.e., $\alpha_u = \alpha_{\max}$, we apply Bayesian optimization to approximate the minimizer of $\phi(\alpha)$. This process continues until a step satisfying the strong Wolfe conditions is found or the number of function evaluations exceeds a predefined limit. If Bayesian optimization does not identify a valid step, we refine the interval using a strategy inspired by the Moré-Thuente update rules and retry Bayesian optimization on the new interval.

Unlike traditional methods that consider only two previous evaluations, Bayesian optimization tracks all function and gradient values observed during the search. We maintain a dataset D of all previously evaluated steps. If at any point a step satisfying the strong Wolfe conditions is found and offers better improvement than prior steps, the line search terminates immediately.

During interval refinement, we define an auxiliary function $\Psi(\alpha)$ to determine the next nested interval. Initially, we set $\Psi = \psi$, but if we encounter a step α_t where $\psi(\alpha_t) \leq 0$ and $\phi'(\alpha_t) > 0$, we switch to $\Psi = \phi$, following the Moré-Thuente approach.

In each iteration, Bayesian optimization selects the next candidate step α_t within (α_l, α_u). If Bayesian optimization fails to improve upon the interval endpoints, we select the step in $\mathcal{D}$ with the highest Gaussian kernel density estimation [30] within the current interval, as Bayesian optimization tends to produce denser observations around minimizers asymptotically [23,24]. Since α_t is now distinct from α_l and α_u, we update the interval according to Moré-Thuente's updating rules (U1-U3):

Case U1: If $\Psi(\alpha_t) > \Psi(\alpha_l)$, then $\alpha_l^+ \leftarrow \alpha_l$ and $\alpha_u^+ \leftarrow \alpha_t$
Case U2: If $\Psi(\alpha_t) \leq \Psi(\alpha_l)$ and $\Psi'(\alpha_t)(\alpha_l - \alpha_t) > 0$, then $\alpha_l^+ \leftarrow \alpha_t$ and $\alpha_u^+ \leftarrow \alpha_u$
Case U3: If $\Psi(\alpha_t) \leq \Psi(\alpha_l)$ and $\Psi'(\alpha_t)(\alpha_l - \alpha_t) < 0$, then $\alpha_l^+ \leftarrow \alpha_t$ and $\alpha_u^+ \leftarrow \alpha_l$

This process is repeated until a step satisfying the strong Wolfe conditions is found or the iteration limit is reached. To ensure finite convergence, we impose a condition on the interval size: if the interval has not decreased by a factor of $\delta < 1$ (typically $\delta = 2/3$) over two consecutive refinements, we force a bisection step. Moré-Thuente show that this process returns a step satisfying the strong Wolfe conditions or α_{max}.

3.3 Bayesian Optimization Implementation

The Bayesian optimization approximates the minimizer of ϕ on an interval $[\alpha_{min}, \alpha_{max}]$ with $\alpha_{min} < \alpha_{max}$. Bayesian optimization runs until a set threshold on the number of condition steps is reached or a step satisfying the strong Wolfe conditions is found. In each iteration, we condition the Gaussian process used as the Bayesian optimization surrogate on all function values and gradients of the steps in $\mathcal{D} \cap [\alpha_{min}, \alpha_{max}]$. Our Gaussian process prior uses the Matérn kernel with $\nu = \frac{5}{2}$ and length scale hyperparameter $|\alpha_{max} - \alpha_{min}|$. The prior mean is a constant at the best-known function value ($\mu(x) = \min\{\phi(\alpha)|\alpha \in \mathcal{D} \cap [\alpha_{min}, \alpha_{max}]\}$), as we have no other information about the structure of the black-box objective function.

We maximize the Lower Confidence Bound (LCB) acquisition function using a hybrid global-local optimizer that employs the DIRECT [14,15] optimizer to find a step and refines it using L-BFGS-B [42]. The new step is added to $\mathcal{D}$ and used to condition the Gaussian process in the next iteration.

If the number of steps used to condition the surrogate Gaussian process exceeds a parameter threshold without identifying a step satisfying the strong Wolfe conditions, we return the step in $\mathcal{D} \cap [\alpha_{min}, \alpha_{max}]$ with the best function value.

3.4 Convergence Guarantee

To prove that the discovery of the initial interval with finite endpoints terminates in a finite number of iterations, we show that the number of interval updates required until a step satisfying the strong Wolfe conditions can be guaranteed to be within the interval or $\alpha_u = \alpha_{max}$ is bounded from above.

Theorem 2. *Let ϕ_{min} be a strict lower bound for ϕ. Let $[\alpha_l = 0, \alpha_u = \alpha_0 = \min(1, \alpha_{max})]$ be the initial interval. The number of interval updates of the form $[\alpha_u, \min(c{\cdot}\alpha_u, \alpha_{max})], c \in \mathbb{R}_{>1}$, until $\alpha_u = \alpha_{max}$ or $\psi(\alpha_u) \geq \psi(\alpha_l)$ or $\psi'(\alpha_u) \geq 0$ is bounded from above by*

$$\left\lceil \frac{1}{\alpha_0} \log_c \left(\min \left(\frac{1}{\mu} \cdot \frac{\phi_{min} - \phi(0)}{\phi'(0)}, \alpha_{max} \right) \right) \right\rceil$$

Proof. For all $\alpha > \alpha_b$ defined as

$$\alpha_b := \frac{1}{\mu} \cdot \frac{\phi_{min} - \phi(0)}{\phi'(0)}$$

we have

$$\begin{aligned}
\psi(\alpha) &= \phi(\alpha) - (\phi(0) + \mu \cdot \phi'(0) \cdot \alpha) \\
&> \phi_{min} - (\phi\,(0) + \mu \cdot \phi'\,(0) \cdot \alpha_b) \\
&= \phi_{min} - \left(\phi(0) + \mu \cdot \phi'(0) \cdot \frac{1}{\mu} \cdot \frac{\phi_{min} - \phi(0)}{\phi'(0)} \right) \\
&= \phi_{min} - \phi_{min} = 0
\end{aligned}$$

This means that no $\alpha > \alpha_b$ can satisfy the sufficient decrease condition. Suppose we update the interval as specified enough, such that $\alpha_u > \alpha_b$, without any prior interval satisfying $\psi(\alpha_u) \geq \psi(\alpha_l)$ or $\psi'(\alpha_u) \geq 0$. Then we have $\alpha_u > \alpha_b$, $\psi(\alpha_l) \leq \psi(0) \leq 0$, $\psi'(\alpha_l) < 0$ and $\psi(\alpha_u) > 0$. Thus, we have found an interval with $\psi(\alpha_u) \geq \psi(\alpha_l)$. The number of interval updates required to reach this interval or have $\alpha_u = \alpha_{max}$ is $\left\lceil \frac{1}{\alpha_0} \log_c \left(\min\left(\alpha_b, \alpha_{max}\right)\right)\right\rceil$. □

Since the interval found using the above process satisfies the invariant of Moré-Thuente up to reordering, the interval refinement process terminates in a finite number of iterations. Combining the convergence guarantee of the initial interval selection and the convergence guarantee of the interval refinement process, we can ensure that our Bayesian line search terminates after a finite number of iterations, returning either a step satisfying the strong Wolfe conditions or $\alpha_{\max}$.

4 Experiments

To evaluate the performance of our Bayesian optimization-based line search method described in Section 3, we integrate it into the GENO solver [19], which employs a quasi-Newton optimization approach, and replace its existing line search component.

We conduct experiments similar to those by Prusina and Laue [28], comparing the performance of our method in terms of convergence, the number of function evaluations, and runtime. In addition to the GENO solver with Bayesian line search (Bayesian GENO), we evaluate:

- The GENO solver with cubic spline interpolation line search from Prusina and Laue [28] (Cubic GENO),
- The L-BFGS-B solver [42] interfaced via SciPy [36], which uses Moré-Thuente line search [25],
- The Ipopt solver, which employs a filter line search [37,38].

While Bayesian GENO and Cubic GENO modify only the line search component of the GENO solver, the Ipopt and L-BFGS-B experiments use entirely different solver implementations. As a result, performance differences may partly reflect solver-specific factors, while still offering useful comparative insights.

The experiments are conducted on 292 unconstrained and 163 bound-constrained problems from the CUTEst test set [12], with a time limit of 3000 seconds per problem. Three unconstrained problems with an unbounded optimal solution are excluded.

The experiments are executed on a machine with an Intel Xeon Gold 5315Y CPU (3.20 GHz), 256 GiB of RAM, and Ubuntu 22.04.4 LTS. All solvers access objective function values and gradients, starting from the same initial point x_0.

The code for our benchmark is available on GitHub[2].

[2] https://github.com/RobinLabryga/bayesian-geno-benchmark/tree/bayesian-ls-paper

4.1 Convergence

A solver is considered converged if it satisfies at least one of the following conditions:

1. Function value convergence, defined as:
$$\frac{f_{solver} - f^\star}{1 + |f^\star|} < 10^{-4} \tag{6}$$
where f_{solver} is the function value obtained by the solver, and f^* is the best function value found by any solver, serving as an approximation of the global optimum.
2. Gradient convergence, defined as:
$$\frac{\|g\|_\infty}{1 + |f_{solver}|} < 10^{-6} \tag{7}$$
where $\|\nabla g\|_\infty$ is the infinity norm of the gradient at the final solution point.

To ensure valid comparisons, any solver that violates bound constraints or returns an inconsistent function value is corrected by clipping the solution to the feasible bounds and recomputing the function value and gradient.

Table 1 presents the number of problems successfully solved by each method for both unconstrained and bound-constrained cases. The best values in each column are highlighted.

Table 1. The number of unconstrained and bound-constrained problems each method solved. Column f conv. shows how many problems each method solved by the function value criterion (Eq. (6)), while column g conv. shows how many problems each solver solved by the gradient criterion (Eq. (7)). Column conv. shows the number of problems where at least one of the two criteria is satisfied. The best value in each column is bold.

	unconstrained			bound-constrained		
Solver	f conv.	g conv.	conv.	f conv.	g conv.	conv
Bayesian GENO	**256**	**222**	**269**	**140**	**145**	**156**
Cubic GENO	252	213	263	125	127	142
Ipopt	195	194	236	124	93	136
L-BFGS-B	232	217	246	135	139	146

Our line search method outperforms all other solvers in all three categories. The Bayesian GENO solver converges on more problems in function value, gradient, or at least one criterion compared to the other solvers for both unconstrained and bound-constrained problems.

4.2 Function Evaluations

Since Bayesian optimization promises a lower function evaluation requirement to approximate a minimizer, we compare the number of function evaluations used by each solver. Figure 2 (a) and (b) compare the solvers regarding the number of function evaluations for unconstrained and bound-constrained problems respectively.

The Bayesian GENO solver requires fewer function evaluations than Ipopt and Cubic GENO to reach the same distance to the optimum $f^\star$. Compared to L-BFGS-B, the Bayesian GENO solver is initially on par but outperforms L-BFGS-B with increasing function evaluations on unconstrained problems. On bound-constrained problems, the Bayesian GENO solver initially requires more function evaluations than L-BFGS-B but outperforms it with increasing function evaluations. This is expected since Bayesian GENO uses a more global exploration approach, requiring more function evaluations per line search. The improvement over Cubic GENO indicates a more efficient exploration strategy in Bayesian GENO.

4.3 Overhead

Each solver incurs different overhead in addition to the time required to evaluate the objective function. We compare the overhead incurred by each solver.

The overhead per function evaluation of a solver for a problem is calculated as

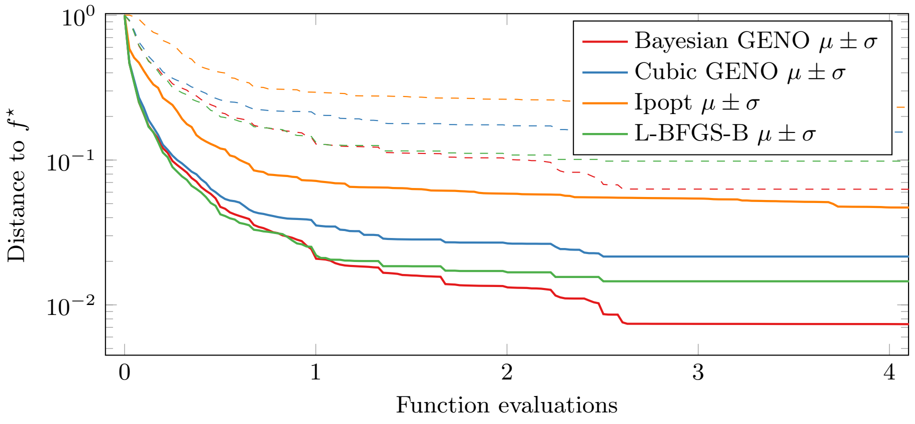
$$\frac{t_{\text{solver}}}{n_{\text{solver}}} - \mathbb{E}[t_{\text{problem}}] \tag{8}$$

where t_{solver} is the total time the solver took to solve the problem, n_{solver} is the number of function evaluations the solver performed, and $\mathbb{E}[t_{\text{problem}}]$ is the expected time to evaluate the objective function once. The expected time $\mathbb{E}[t_{\text{problem}}]$ is estimated by measuring the runtime of 1000 function evaluations and taking the average. The box plots in Fig. 3 depict the range of overhead estimations across the problems for each solver.

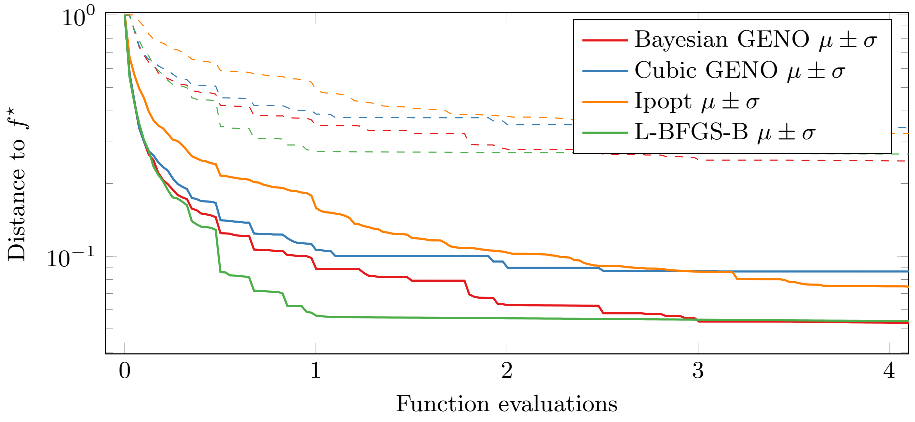
The Bayesian GENO solver has a higher overhead per function evaluation than the Cubic GENO and L-BFGS-B solvers. The overhead of Bayesian GENO is comparable to that of Ipopt. The added complexity of Gaussian process regression compared to cubic spline interpolation may explain the performance difference between Bayesian GENO and Cubic GENO. This is supported by the increased overhead in the bound-constrained setting compared to the unconstrained setting, where Bayesian optimization may search for a non-existent step satisfying the strong Wolfe conditions whenever the search interval includes α_{max}.

5 Conclusion

We have introduced a novel line search method that uses Bayesian optimization to better utilize information from function and gradient evaluations. Our

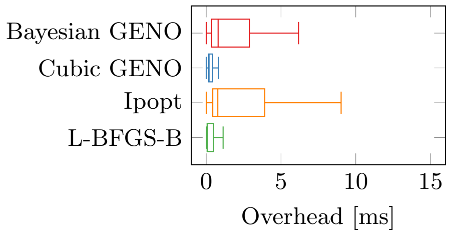


(a) Unconstrained problems

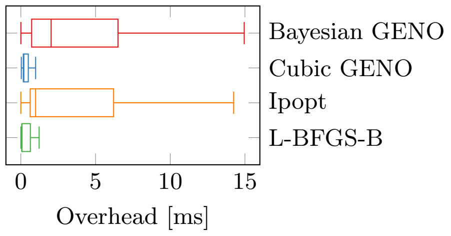


(b) Bound-constrained problems

Fig. 2. A comparison between the number of function evaluations required by each solver in the experiments. For each problem, we calculate the distance of the best-observed function value to $f^\star$ at every function evaluation. We then normalize the distance to $f^\star$ such that the distance at the first function evaluation is 1.0. Similarly, we normalize the number of function evaluations such that the number of function evaluations at the discovery of $f^\star$ is 1.0. We then calculate the mean (solid lines) and standard deviation (dashed lines) of the normalized distances across all problems. The x-axis shows the number of function evaluations, while the y-axis shows the distance to $f^\star$.



(a) Unconstrained problems



(b) Bound-constrained problems

Fig. 3. Box plots depicting the overhead per function evaluation in milliseconds for each solver on the unconstrained and bound-constrained problems computed as in Eq. (8).

approach addresses the limitations of traditional line search techniques, which discard valuable information during iterative refinement. By employing a Gaussian process surrogate model, our method finds more informed step lengths, offering a solution that is both theoretically sound and practically efficient.

Through extensive empirical evaluations on the challenging unconstrained and bound-constrained optimization problems from the CUTEst test set, we demonstrated the superiority of the proposed method when integrated into the GENO solver. The Bayesian line search outperformed state-of-the-art methods in terms of convergence on both function value and gradient criteria, solving a greater number of problems across diverse problem types. While it incurs higher computational overhead, this method is especially valuable in optimization scenarios where function evaluations are expensive.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Armijo, L.: Minimization of functions having lipschitz continuous first partial derivatives. Pacific Journal of mathematics **16**(1), 1–3 (1966)
2. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA journal of numerical analysis **8**(1), 141–148 (1988)
3. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599 (2010)
4. Cox, D., John, S.: A statistical method for global optimization. In: IEEE International Conference on Systems, Man, and Cybernetics. pp. 1241–1246 vol.2 (1992)
5. Dennis Jr, J.E., Schnabel, R.B.: Numerical methods for unconstrained optimization and nonlinear equations. SIAM (1996)
6. Fiedler, C., Scherer, C.W., Trimpe, S.: Practical and rigorous uncertainty bounds for gaussian process regression. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 7439–7447 (2021)

7. Forrester, A., Sobester, A., Keane, A.: Engineering design via surrogate modelling: a practical guide. John Wiley & Sons (2008)
8. Frazier, P., Powell, W., Dayanik, S.: The knowledge-gradient policy for correlated normal beliefs. INFORMS journal on Computing **21**(4), 599–613 (2009)
9. Frazier, P.I.: Bayesian optimization. In: Recent advances in optimization and modeling of contemporary problems, pp. 255–278. Informs (2018)
10. Frazier, P.I.: A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811 (2018)
11. Giesen, J., Laue, S.: Combining ADMM and the augmented lagrangian method for efficiently handling many constraints. In: International Joint Conference on Artificial Intelligence (IJCAI) (2019)
12. Gould, N.I., Orban, D., Toint, P.L.: Cutest: a constrained and unconstrained testing environment with safe threads for mathematical optimization. Computational optimization and applications **60**, 545–557 (2015)
13. Hager, W.W.: A derivative-based bracketing scheme for univariate minimization and the conjugate gradient method. Computers & Mathematics with Applications **18**(9), 779–795 (1989)
14. Jones, D.R., Martins, J.R.: The direct algorithm: 25 years later. Journal of global optimization **79**(3), 521–566 (2021)
15. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the lipschitz constant. Journal of optimization Theory and Applications **79**, 157–181 (1993)
16. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization **13**, 455–492 (1998)
17. Krige, D.G.: A statistical approach to some basic mine valuation problems on the witwatersrand. Journal of the Southern African Institute of Mining and Metallurgy **52**(6), 119–139 (1951)
18. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise (1964)
19. Laue, S., Mitterreiter, M., Giesen, J.: Geno – generic optimization for classical machine learning. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)
20. Laue, S.: On the equivalence of automatic and symbolic differentiation. arXiv preprint arXiv:1904.02990 (2022)
21. Laue, S., Blacher, M., Giesen, J.: Optimization for classical machine learning problems on the GPU. In: Conference on Artificial Intelligence (AAAI) (2022)
22. Mahsereci, M., Henning, P.: Probabilistic line searches for stochastic optimization. Journal of Machine Learning Research **18**(119), 1–59 (2012)
23. Mockus, J.: Application of bayesian approach to numerical methods of global and stochastic optimization. Journal of Global Optimization **4**, 347–365 (1994)
24. Mockus, J.B., Mockus, L.J.: Bayesian approach to global optimization and application to multiobjective and constrained problems. Journal of optimization theory and applications **70**, 157–172 (1991)
25. Moré, J.J., Thuente, D.J.: Line search algorithms with guaranteed sufficient decrease. ACM Trans. Math. Softw. **20**(3), 286–307 (sep 1994)
26. Nocedal, J., Wright, S.J.: Numerical optimization (2006)
27. Papageorgiou, D.J., Kronqvist, J., Kumaran, K.: Linewalker: line search for black box derivative-free optimization and surrogate model construction. Optimization and Engineering pp. 1–65 (2024)
28. Prusina, T., Laue, S.: Efficient line search method based on regression and uncertainty quantification. In: Learning and Intelligent Optimization (LION) (2024)

29. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. Adaptive computation and machine learning, MIT Press (2006)
30. Scott, D.W.: Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons (2015)
31. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)
32. Sra, S., Nowozin, S., Wright, S.J.: Optimization for Machine Learning. MIT Press (2012)
33. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.: Gaussian process optimization in the bandit setting: No regret and experimental design. arXiv preprint arXiv:0912.3995 (2009)
34. Stuckman, B.: A global search method for optimizing nonlinear systems. IEEE Transactions on Systems, Man, and Cybernetics **18**(6), 965–977 (1988)
35. Vazquez, E., Bect, J.: Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. Journal of Statistical Planning and Inference **140**(11), 3088–3095 (2010)
36. Virtanen, P., Gommers, R., Oliphant, T., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al.: Fundamental algorithms for scientific computing in python and scipy 1.0 contributors. scipy 1.0. Nat. Methods **17**, 261–272 (2020)
37. Wächter, A., Biegler, L.T.: Line search filter methods for nonlinear programming: Motivation and global convergence. SIAM Journal on Optimization **16**(1), 1–31 (2005)
38. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming **106**, 25–57 (2006)
39. Wolfe, P.: Convergence conditions for ascent methods. SIAM review **11**(2), 226–235 (1969)
40. Wolfe, P.: Convergence conditions for ascent methods. ii: Some corrections. SIAM review **13**(2), 185–188 (1971)
41. Wu, J., Poloczek, M., Wilson, A.G., Frazier, P.: Bayesian optimization with gradients. Advances in neural information processing systems **30** (2017)
42. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on mathematical software (TOMS) **23**(4), 550–560 (1997)