Non-Myopic Bayesian Optimization with Unknown Cost Constraints

Ethan Hersch[*]
Department of Computer Science
Cornell University
Ithaca, NY 14853
esh87@cornell.edu

Darian Nwankwo[*]
Department of Computer Science
Cornell University
Ithaca, NY 14853
don4@cornell.edu

Abstract

Bayesian Optimization (BO) is a powerful tool for optimizing expensive black-box functions. It treats the objective function as a probabilistic model and iteratively updates this model based on observations, typically using Gaussian Processes (GPs). Bayesian optimization can be used in a setting where evaluation is expensive, say tuning the parameters of a neural network to optimize accuracy. Consider the scenario where each evaluation point has a cost associated to it, and there is a budget of the total cost to optimize the underlying function. Say this cost is unknown a-priori and it must be explored as the objective function is evaluated. This is the setting this paper will explore and propose a methodology to model such a scenario by building a GP around both the cost function and objective function.

This paper is an adaptation of a submission by Ethan and Darian for project for CS 6784. I highlight my own contributions and learning from this project and my hopes for future research in this area.

1 Introduction

Bayesian optimization (BO) algorithms are sample-efficient methods for global optimization of expensive continuous, non-convex "black-box" functions where derivative information is not available. We assume that these functions are defined over a compact set where evaluation is expensive. BO builds a statistical model of the objective function based on samples. BO further builds an acquisition function, conditioned on the statistical model, and chooses new sample locations by optimizing this acquisition function that reflects the value of candidate locations. Most BO methods use *myopic* acquisition functions that only consider the impact of the next sample where every sample has unit cost. Given the aforementioned description, our objective is the following:

$$x^* \in \arg\min_{x \in \mathcal{X}} f(x)$$

Where f is our objective function to minimize, $\mathcal{X}$ is our domain over which we will collect observations and x is a particular observation. The de facto statistical model used in BO is a Gaussian process, i.e. we place a Gaussian process prior over our function $f \sim \mathcal{GP}(\mu(x), k(x, x'))$. Here $\mu(x)$ is our predictive mean at an observation x and $k(x, x')$ is the kernel, or covariance matrix, between our observation x and some new observation x'. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [6]. Traditional BO methods assume the cost of evaluation is uniform across our domain, which is implicitly encoded in most acquisition functions. A common myopic acquisiton functions is expected improvement (EI) [A.1.1].

[*]These authors contributed equally to this work

Preprint. Under review.

In practice, the cost of evaluation differs across our domain $\mathcal{X} \subseteq \mathbb{R}^d$ and standard acquisition functions do not consider this. Cost-constrained BO considers other acquisition functions, such as expected improvement per unit cost ($EIpu$) [A.1.2] or a constrained Markov decision process ($CMDP$) [A.1.3] to weigh the cost of evaluations with the expected value of said evaluation. $CMDP$ tends to outperform $EIpu$ because it is *non-myopic* and has a more global outlook on the result of cost on the total optimization routine. BO's primary strength lies in its sample efficiency of evaluating the objective, making it well-suited for applications where cost constraints become crucial.

We contribute a novel approach to BO in a stochastic cost setting. We model cost and objective with a GP for a dual-GP setup and use a non-myopic acquisition function. Unknown cost settings have not been extensively explored, and it has not been explored in a non-myopic setting with dual-GPs.

2 Background and previous work

Traditional BO methods focus on one-step look-ahead acquisition functions, such as Expected Improvement (EI) [A.1.1], which optimize immediate gain and do not consider future evaluations. Multi-step look-ahead acquisition functions have emerged, aiming to maximize the cumulative reward across evaluations. These methods balance exploration against exploitation over a finite budget.

A prominent solution in multi-step look-ahead BO formulated the selection process as a dynamic programming problem solved throughout roll-out strategies [2]. However, the computational expense of this roll-out strategy limited the scalability of the approach. In response, [10], introduced 2-OPT-C which was a constrained two-step acquisition function utilizing likelihood ratios for efficient optimization and balances exploration. Practical advancements, such as the Monte Carlo and variance reduction techniques presented in the 2-step look-ahead algorithm by Wu and Frazier have reduced this computational barrier making the non-myopic BO approach feasible in higher dimensions [9].

[4] tackled the cost-constrained aspect of BO by framing it as a constrained Markov decision process ($CMDP$), where the costs are deterministic and depend on the region within the search space. Their approach uses a roll-out approximation to the optimal $CMDP$ policy by taking the cost and future iterations into account. This model is effective for applications where evaluation costs vary widely, such as hyper-parameter tuning, where cost efficiency is more critical than sample efficiency.

3 Research question and approach

One key limitation in existing non-myopic BO is the assumption that the cost of each evaluation is known in advance. This is often not the case in practice, where the cost of evaluation may not be fully known a-priori. The need for a cost-aware optimization framework that can handle such scenarios is evident. One can think of cost-constrained BO as the following constrained optimization problem:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \,\text{subject to}\, \sum_{\mathbf{x}\in\mathcal{X}} c(\mathbf{x}) \leq \tau.$$

We assume $f(\mathbf{x})$ outputs not only its value, but also its evaluation cost determined by a black-box cost function $c(\mathbf{x}) > 0$. We assume the cost function has the following functional form: $c(\mathbf{x}) = p(\mathbf{x}) + \exp(g(\mathbf{x}))$, where $g(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')$ and $p(\mathbf{x}) = \sum_{n=0}^{\infty} a_n(x - x_0)^n > 0$. That is, our cost has some optional known analytic component plus some perturbation term that is state-independent. This means that the cost function will have to now also be modeled by a GP. So we need a handle on the predictive mean and variance of our cost. Our goal is to minimize $f(\mathbf{x})$ subject to the total evaluation cost not exceeding τ. The value of τ is a pre-determined upper bound on the total cost, such as compute time, dollars, or cloud-compute resource utilization.

In such tasks, each evaluation can be expensive, requiring a method that balances exploration and exploitation. The classical formulation of BO assumes a "myopic" view of decision-making. Instead, a non-myopic (NMBO) formulation introduces a new paradigm that extends the utility of BO. Lam et al. [3] formulate NMBO as a finite horizon dynamic program [A.5]. The objective is to find an optimal policy for the h-horizon MDP that maximizes the expected reward given some decision.

Our work builds on these previous advancements within cost-constrained BO by proposing a methodology that models the cost functions within a GP framework, addressing scenarios where cost functions are unknown and must be dynamically explored alongside the objective function. Simultaneously, we incorporate our stochastic model of the cost function into the cost-constrained

Markov decision process to maintain our balance of exploration against exploitation. Hence, **how can we perform non-myopic cost- constrained Bayesian optimization in a setting where the cost function is not known ahead of time?**

4 Experiments and results

The Python code to model these experiments can be found in this repository link `https://github.coecis.cornell.edu/esh87/Stochastic-Cost-NMBO/`. There are currently few figures in this repository as it was imported from a Google Colab, but the significant figures are in the appendix. Throughout this section, the figures refered to can be found in Appendix A.7.

To motivate our experiments, we selected standard test optimization functions as our "datasets." The first notable function is the Ackley function [A.2.1], which is useful as it exhibits many local minima. Other functions include the Branin [A.2.3] (which has three global minima) and Rosenbrock function [A.2.2], which is valley-shaped. The last two functions used are the 2D-Beale [A.2.4], which has sharp corners at the edge of its $[-4.5, 4.5]$ domain, and Gramacy & Lee function [A.2.5], which is a simple cyclic optimization function. These standard optimization functions are found from Simon Fraser University [8]. Our initial experiments deal with the Ackley function in one dimension as a stress test to ensure our dual-GP setup works.

We used BOTorch and GPyTorch to set up machinery for fitting GPs around both the cost and objective (Ackley) function. Figure 1 shows the dual surrogate fit around these initial samples. We then compared random sampling (a non-BO technique) with the $EIpu$ and EI acquisition functions to show how Bayesian optimization converges to a gap of 1 sooner (random sampling does not converge to a gap of 1). Gap is our loss metric: a gap of 1 indicates our BO algorithm has found the optimal solution. The goal in BO is to obtain a gap of 1 sooner than other optimization objectives [A.4]. See Figure 2 for this demonstration. A steeper rise in the curve indicates that the global minimum is being found quicker, while accumulating less cost. Thus, the better the acquisition function, the steeper its slope is in these figures. In this example, consider cost to just be the number of iterations, but in future tests, we will use a synthetic cost function.

These results are important as they show we have set up the machinery to represent both cost and our objective as a probabilistic function. They also show baseline performance of standard myopic BO techniques. Figure 2 shows that $EIpu$ achieves the fastest convergence, followed by EI and lastly with random sampling. These are our baseline stress tests to build future work off of. In the latter portion of our work, we set up the $CMDP$ acquisition function to get a handle on non-myopic BO and use these dual GPs to consider the predictive mean of our cost, rather than the pure cost.

We propose a novel acquisition function, TwoStepExpectedImprovementWithCost using $CMDP$ in the stochastic cost setting [A.6].

The process begins by evaluating the immediate acquisition for each candidate point x. We do this by computing the Expected Improvement from the current GP model and normalizing it by the predicted cost raised to the power of α. Next, the algorithm computes the second-step acquisition by using Monte Carlo simulations to account for the uncertainty in model predictions at x. Specifically, posterior samples are drawn from the GP model at x, representing potential outcomes if x were evaluated. For each simulated outcome, the algorithm fits a temporary GP model, augmented with the simulated observation, and calculates the EI and associated cost for this second-step evaluation.

We average the acquisition values for all second-step scenarios to provide an estimate of the expected future improvement at x. The total acquisition value combines the immediate and second-step components, guiding the surrogate model toward evaluations that optimize improvement while managing costs. By leveraging Monte Carlo simulations to simulate possible outcomes and anticipate future gains, our algorithm achieves a more robust and resource-aware optimization strategy.

We now test this algorithm's performance on synthetic objective functions and various synthetic quadratic cost functions [A.3].

The Knowledge Gradient (KG) acquisition function serves as a baseline to our approach [A.1.4]. KG focuses on estimating how much evaluating a candidate point will improve the model's ability to make better decisions in subsequent steps, maximizing the value of new information obtained.

See A.7.2 for figures of the 2-step look-ahead algorithm's performance. These plots test 2-step look-ahead against $EIpu$ and KG on various 2D objective and 1D synthetic cost functions (see A.3 for an explanation of the cost functions). We generate these curves in expectation over 10 runs as random initial seeds (whose size varies between experiments) cause the BO algorithm results to differ between experiments. We observe that in many instances, our method converges to a gap of 1.0 at a faster rate than both the knowledge gradient and $EIpu$. This can be seen by examining how much cost is accumulated over specific measurements of the gap. The tests on the Ackley and Rosenbrock functions with quadratic cost demonstrate the advantage of our method [Figures 3 and 4]

Figures 5, 6, and 7 do not converge to a gap of 1.0, but the 2-step look-ahead curve is steeper, which indicates it achieves globally optimum solutions by accumulating less cost. Generally (this is more pronounced with the Ackley and Rosenbrock functions), the 2-step look-ahead curves display less standard error than $EIpu$, meaning they have less variance between initial seeds and more predictable behavior. This means our approach also has more predictable behavior than others.

Dataset	2-step look-ahead	Knowledge Gradient
2D Ackley	31.6%	11.8%
2D Rosenbrock	7.2%	4.4% (to a gap of 0.99)
2D Branin	22.1% (to a gap of 0.95)	N/A (never achieves a gap of 0.95)

Table 1: Percentage accumulated cost improvement over $EIpu$

These three experiments are a sound example of the performance of our method over a myopic $EIpu$ and baseline non-myopic approach using the knowledge gradient. The knowledge gradient offers improvement over $EIpu$ on some datasets, and our 2-step look-ahead approach offers considerable gains over $EIpu$ as well. Overall, when sampling on different objective and cost functions, and varying the initial seed, our method yields faster gap convergence accumulating lower cost.

Figures 8 and 9 are plots where the z-axis represents the amount of cost for each sample in the BO loop for each iteration. These plots were also taken in expectation over 10 runs. Each of these plots demonstrates that both the 2-step look-ahead and $EIpu$ methods prioritize sampling low-cost regions initially before transitioning to high-cost regions. This behavior directly illustrates the exploration-exploitation trade-off. At the beginning, our algorithm focuses on low-cost samples to explore the search space. Once it identifies a region that is more likely to contain the global minimum, it shifts to exploring higher-cost regions. While the myopic and non-myopic approaches exhibit this pattern, the myopic $EIpu$ method occasionally revisits low-cost regions even after sampling high-cost ones.

5 Discussion and conclusion

We proposed a 2-step look-ahead algorithm that demonstrates promising results in the unknown cost setting. By comparing this algorithm to a standard myopic cost-aware acquisition function and a common non-myopic one (KG), we observe that our method generally achieves faster convergence to the global minimum while incurring the least cost.

Future research could explore extending the algorithm to greater look-ahead steps to evaluate how performance evolves, though this is currently limited by computational resources. With more time and resources, we could also experiment more with scalability in functions in greater dimensions. Additionally, parallel programming could be leveraged to accelerate multi-step look-ahead computations, making such approaches more feasible.

Another promising direction is applying cost-constrained BO to hyperparameter tuning for neural networks, where balancing cost and performance is crucial. This is an ideal setting as each combination of hyperparameters produces a different accuracy of the network, but training times also vary. By leveraging and algorithm with smart, efficient sampling of hyperparamreters, a neural network can be tuned to optimal accuracy while incurring as little cost as possible to get to this combination of hyperparameters. Our approach to cost-aware BO accumulates less cost than others, so it is well-suited for this task.

This project provided me with significant exposure to non-myopic optimization, stochastic cost optimization, and the knowledge gradient acquisition function. I gained valuable experience implementing and experimenting with BOTorch and GPyTorch, which will be useful for future research

projects. Last semester, I focused on implementing a Bayesian Optimization package from scratch in Julia. This semester, by tackling a more complex problem setup and working with numerous datasets, I was able to supplement my theoretical understanding of BO with practical skills for running experiments and testing custom acquisition functions.

The method we proposed, though inspired by previous work, showed promising results in this new stochastic cost setting. This project not only deepened my understanding of BO but also enhanced my ability to conduct hands-on research. It offered valuable experience with a subset of BO that I find particularly compelling and helped me become more comfortable with the research process as a whole.

References

[1] Donald R Jones, Matthias Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions," , vol. 13, no. 4, pp. 455-492, 1998. *Journal of Global Optimization*, 13:455–492, 1998.

[2] Remi Lam and Karen Willcox. Lookahead bayesian optimization with inequality constraints. 2017. URL `https://papers.nips.cc/paper_files/paper/2017/file/83f97f4825290be4cb794ec6a234595f-Paper.pdf`.

[3] Remi R. Lam, Karen E. Willcox, and David H. Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Advances in Neural Information Processing Systems*, (Nips):883–891, 2016.

[4] David Eriksson Lee, Valerio Valerio Perrone, and Matthias Seeger. A nonmyopic approach to cost-constrained bayesian optimization. *arXiv preprint arXiv:2106.06079*, 2021. URL `https://arxiv.org/pdf/2106.06079`.

[5] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[6] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[8] Simon Fraser University. Optimization test functions and datasets. `https://www.sfu.ca/~ssurjano/optimization.html`, 2023. Accessed: 2024-10-28.

[9] Jian Wu and Peter Frazier. Practical two-step lookahead bayesian optimization. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019. URL `https://papers.nips.cc/paper_files/paper/2019/hash/2e6d9c6052e99fcdfa61d9b9da273ca2-Abstract.html`.

[10] Miao Zhang, Huiqi Li, and Steven Su. High dimensional bayesian optimization via supervised dimension reduction. *arXiv preprint arXiv:1907.08953*, 2019. URL `https://arxiv.org/abs/1907.08953`.

A Appendix / supplemental material

A.1 Common Acquisition Functions

A.1.1 Expected improvement

The expected improvement acquisition function ($EI(x)$) is a common acquisition function for BO.

$$EI(x) = (f^* - \mu(x))\Phi\left(\frac{f^* - \mu(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{f^* - \mu(x)}{\sigma(x)}\right)$$

where f^* is the best point observed so far and $\mu(x)$, $\sigma(x)$ are the predictive mean and variance at our point x. Φ is the standard normal CDF and ϕ is the PDF. The expected improvement acquisition function is powerful as it uses statistics gathered on all prior observations to guide the model where it has the "best chance" of finding a value of f which is higher than any previously seen.

A.1.2 Expected Inprovement Per Unit Cost ($EIpu$)

The expected improvement acquisition function is powerful as it uses statistics gathered on all prior observations to guide the model where it has the "best chance" of finding a value of f which is higher than any previously seen. This acquisition function is called $EIpu$:

$$EIpu(x) = \frac{EI(x)}{c(x)}$$

where c is our cost function.

The $EIpu$ acquisition function brings a notable challenge: this is a myopic approach—our acquisition function will naively focus on minimizing the cost function rather than selecting our next point $x^* = \text{argmax}_{x^*} EIpu(x)$ which is a useful observation for also optimizing f.

A.1.3 Constrained Markov Decision Process

This is a non-myopic acquisition function for cost-constrained BO. It is proposed in the Lee et al. paper [4]. The cost function C in $CMDP$ uses a cumulative cost function

$$C_h^\pi(s_0) = \mathbb{E}[\sum_{t=0}^{h-1} C(s_t, \pi_t(s_t), s_{t+1})]$$

Where C_h^π measures total expected cost given a policy π, starting state s_0 and horizon h. In our proposed solution, we use a two-step look-ahead, so $h = 2$.

A.1.4 Knowledge Gradient (KG)

A non-myopic decision-making approach that evaluates the value of acquiring additional information to improve future outcomes. It is often used as a baseline to compare with other strategies, such as the 2-step look-ahead. While the two-step look-ahead focuses on the next two steps, KG considers longer-term impacts, making it useful for understanding the trade-offs between short-term decisions and long-term benefits.

A.2 Synthetic test functions

A.2.1 Ackley

The Ackley function exhibits properties that make finding the global minimizer hard to find for non-gradient-based methods:

$$f(x) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(c\,x_i)\right) + a + \exp(1)$$

where $a = 20$, $b = 0.2$, $c = 2\pi$, $d =$ number of dimensions. The Ackley function is useful to study because it displays frequent local minima but one distinct global minima.

A.2.2 Rosenbrock

The Rosenbrock function is defined as:

$$f(x) = \sum_{i=1}^{d-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right]$$

The function is unimodal, with a global minimum located inside a long, narrow, parabolic-shaped flat valley. This makes optimization challenging as the global minimum is not aligned with the gradient. This valley shape makes convergence to the global minimum difficult.

A.2.3 Branin

The Branin function is a classic two-dimensional benchmark for optimization problems, defined as:

$$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

where $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. It has three global minima and is commonly used to evaluate the ability of optimization methods to find multiple optima.

A.2.4 Beale

The Beale function is another classic two-dimensional test function, defined as:

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2$$

It has a single global minimum at $(x, y) = (3, 0.5)$ and a highly non-linear landscape that makes it challenging for optimization methods. It displays sharp peaks at the corners of its common search domain defined on $[-4.5, 4.5]$ for all x_i.

A.2.5 Gramacy & Lee

The Gramacy & Lee function is a one-dimensional test function often used for studying surrogate modeling and Bayesian optimization. It is defined as:

$$f(x) = x_1 \exp(-x_1^2 - x_2^2)$$

where $x_i \in (-2, 6)$. This function features non-convex behavior with several local optima, making it an excellent test case for evaluating optimization techniques.

A.3 Synthetic Cost Functions

We test our 2-step look-ahead algorithm's performance on synthetic objective functions and various synthetic cost functions. All of these are strictly positive as we do not deal with settings of negative cost.

A.3.1 Quadratic Cost

$$c_1(x) = 200(1.1 - x_1 - x_2)^2$$

This is a simple 2D quadratic cost function.

A.3.2 Exponential Cost

$$c_2(x) = e^{-x_1 - x_2} + 1$$

This is a simple 2D exponential cost function.

A.3.3 Trigonometric Cost

$$c_3(x) = \sin(x_1)^2 + \cos(x_2)^2 + 1$$

This is a simple 2D trigonometric cost function.

A.4 Gap Metric

Given a limited evaluation budget, the performance of an algorithm can be measured in terms of its gap. The gap measures the best decrease in objective function from the first to the last iteration, normalized by the maximum reduction possible.

$$G_i = \frac{f_{min}^{\mathcal{D}_1} - f_{min}^{\mathcal{D}_i}}{f_{min}^{\mathcal{D}_1} - f(\mathbf{x}^*)}$$

Our BO algorithm is at convergence when the gap approaches 1, meaning our current observation is close to the optimal one.

A.5 Non-myopic Bayesian optimization as a finite horizon dynamic program

The notation used is standard (see Puterman [5]): an MDP is a collection $(T, \mathbb{S}, \mathbb{A}, P, R)$, where $T = \{1, 2, \ldots, h\}$, and $h < \infty$ is the set of decision epochs, finite for our problem. The state space, $\mathbb{S}$, encapsulates the information needed to model the system from time $t \in T$, and $\mathbb{A}$ is the action space. Given a state $s \in \mathbb{S}$ and an action $a \in \mathbb{A}$, $P(s'|s, a)$ is the probability the next state will be s'. $R(s, a, s')$ is the reward received for choosing action a in state s and transitioning to s'.

A decision rule $\pi_t : \mathbb{S} \to \mathbb{A}$ maps states to actions at time t. A policy π is a series of decision rules $\pi = (\pi_1, \pi_2, \ldots, \pi_h)$, one at each decision epoch. Given a policy π, a starting state s_0, and horizon h, we can define the expected total reward $V_h^\pi(s_0)$ as:

$$V_h^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{h-1} R(s_t, \pi_t(s_t), s_{t+1})\right].$$

Our objective is to find the optimal policy π^* that maximizes the expected total reward, i.e., $\sup_{\pi\in\Pi} V_h^\pi(s_0)$, where Π is the space of all admissible policies.

If we can sample from the transition probability P, we can estimate the expected total reward of any base policy — the decisions made using the base acquisition function — with MC integration [7]:

$$V_h^{\hat{\pi}}(s_0) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{h-1} R(s_t^i, \hat{\pi}_t(s_t^i), s_{t+1}^i)\right].$$

Given a GP prior over data $\mathcal{D}_t$ with mean $\mu^{(t)}$ and covariance matrix $K^{(t)}$, we model h steps of BO as an MDP. This MDP's state space is all possible data sets reachable from starting-state $\mathcal{D}_t$ with h steps of BO. Its action space is Ω; actions correspond to sampling a point in Ω. Its transition probability and reward function are defined as follows. Given an action x^{t+1}, the transition probability from $\mathcal{D}_t$ to $\mathcal{D}_{t+1}$, where $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\mathbf{x}^{t+1}, y_{t+1})\}$ is:

$$P(\mathcal{D}_t, \mathbf{x}^{t+1}, \mathcal{D}_{t+1}) \sim \mathcal{N}(\mu^{(t)}(\mathbf{x}^{t+1}; \mathcal{D}_t), K^{(t)}(\mathbf{x}^{t+1}, \mathbf{x}^{t+1}; \mathcal{D}_t)).$$

Thus, the transition probability from $\mathcal{D}_t$ to $\mathcal{D}_{t+1}$ is the probability of sampling y_{t+1} from the posterior $\mathcal{GP}(\mu^{(t)}, K^{(t)})$ at $\mathbf{x}^{t+1}$. We define a reward according to expected improvement (EI) [1]. Let f_t^* be the minimum observed value in the observed set $\mathcal{D}_t$, i.e. $f_t^* = \min\{y_0, \ldots, y_t\}$. Then our reward is expressed as follows:

$$R(\mathcal{D}_t, \mathbf{x}^{t+1}, \mathcal{D}_{t+1}) = (f_t^* - f_{t+1})^+ \equiv \max(f_t^* - f_{t+1}, 0).$$

EI can be defined as the optimal policy for horizon one, obtained by maximizing the immediate reward:

$$\pi_{EI} = \arg\max_{\pi} V_1^\pi(\mathcal{D}_t) = \arg\max_{\mathbf{x}^{t+1}\in\Omega} \mathbb{E}\left[(f_t^* - f_{t+1})^+\right] \equiv \arg\max_{\mathbf{x}^{t+1}\in\Omega} EI(\mathbf{x}^{t+1}|\mathcal{D}_t),$$

where the starting state is $\mathcal{D}_t$—our initial samples. We define the non-myopic policy as the optimal solution to an h-horizon MDP. The expected total reward of this MDP is:

$$V_h^\pi(\mathcal{D}_n) = \mathbb{E}\left[\sum_{t=n}^{n+h-1} R(\mathcal{D}_t, \pi_t(\mathcal{D}_t), \mathcal{D}_{t+1})\right] = \mathbb{E}\left[\sum_{t=n}^{n+h-1} (f_t^* - f_{t+1})^+\right] = \mathbb{E}\left[\left(f_n^* - \min_{t\le n+h} f_t\right)^+\right].$$

The integrand is a telescoping sum, resulting in the equivalent expression on the rightmost side of the equation. When $h > 1$, the optimal policy is difficult to compute.

A.6 Algorithms

Algorithm 1 TwoStepExpectedImprovementWithCost

Input: GP model f_{GP}, cost model CostGP, exploration factor α, bounds $\mathcal{B}$
Initialize: Immediate acquisition $a_{\text{immediate}}(x)$ and second-step acquisition $a_{\text{second}}(x)$
for each $x \in \mathcal{B}$ **do**
    $EI_{\text{immediate}}(x) \leftarrow \text{ExpectedImprovement}(f_{\text{GP}}, x)$
    $c_{\text{immediate}}(x) \leftarrow \text{CostGP}(x)$
    $a_{\text{immediate}}(x) \leftarrow \frac{EI_{\text{immediate}}(x)}{c_{\text{immediate}}(x)^{\alpha}}$
    Simulate posterior at x: $Y_{\text{posterior}} \sim \text{Posterior}(f_{\text{GP}}, x)$
    for each $y \in Y_{\text{posterior}}$ **do**
        Fit temporary GP models with $(X \cup x, Y \cup y)$
        Compute $EI_{\text{second}}(x)$ and $c_{\text{second}}(x)$
        $a_{\text{second}}(x) \leftarrow a_{\text{second}}(x) + \frac{EI_{\text{second}}(x)}{c_{\text{second}}(x)^{\alpha}}$
    end for
    Average $a_{\text{second}}(x)$ over simulations
end for
Output: Acquisition value $a(x) = a_{\text{immediate}}(x) + a_{\text{second}}(x)$

A.7 Figures

A.7.1 Stress Test Figures

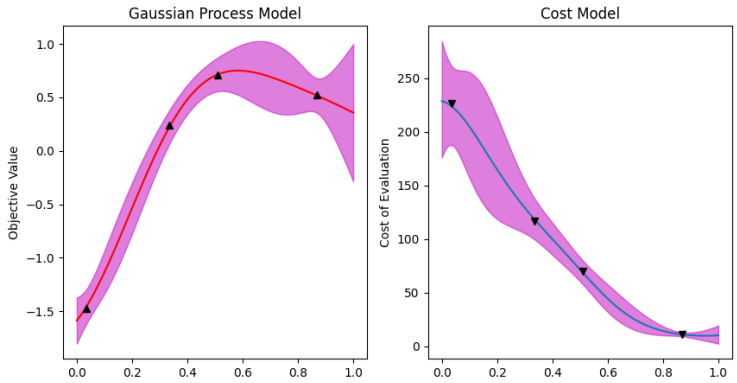

Figure 1: Initial surrogate model

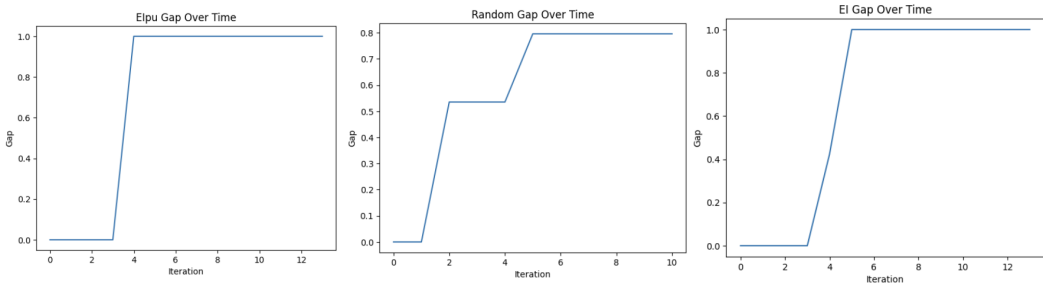

Figure 2: Comparison of gap across optimization techniques

A.7.2 Expected gap comparison plots

These plots test the performance of our 2-step look-ahead approach against other baselines: knowledge gradient and $EIpu$. These plots are generated in expectation over 10 runs to account for the differences in BO performance with different initial seeds. Each figure displays a different synthetic 2D test function on the domain $[-5, 10]$ with a synthetic strictly-positive cost function. The plots measure expected gap against the cumulative cost (cumulated on this synthetic cost function) as the cost-aware BO algorithm runs. Some samples do not achieve a gap of 1.0 but get close. This limitation was imposed because running more than 10 iterations caused our Google Colab environment to crash, but 10 iterations are sufficient to reveal the behavior of each method in most experiments.

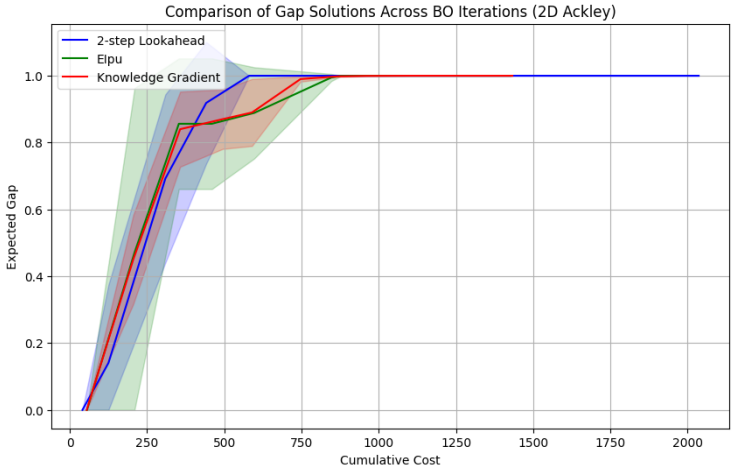

Figure 3: 2D Ackley with 4 initial seeds and quadratic cost

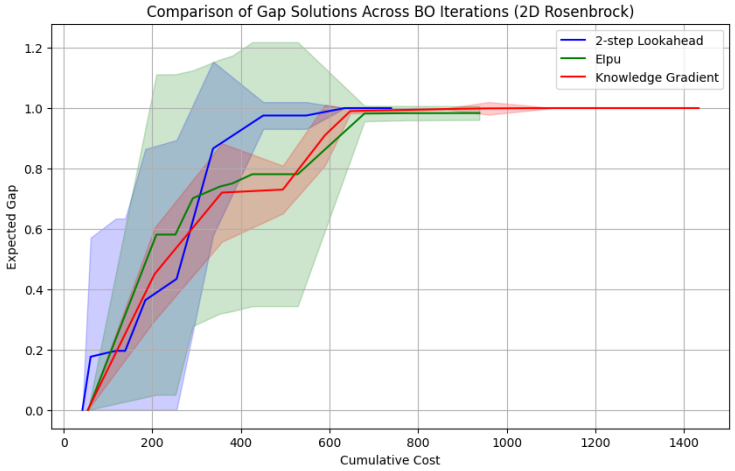

Figure 4: 2D Rosenbrock with 4 initial seeds and quadratic cost

Here, the blue line is slightly above the others. This indicates 2-step look-ahead finds a solution closer to the global minimum than the other acquisition functions. These are still minor differences in minima approximation.

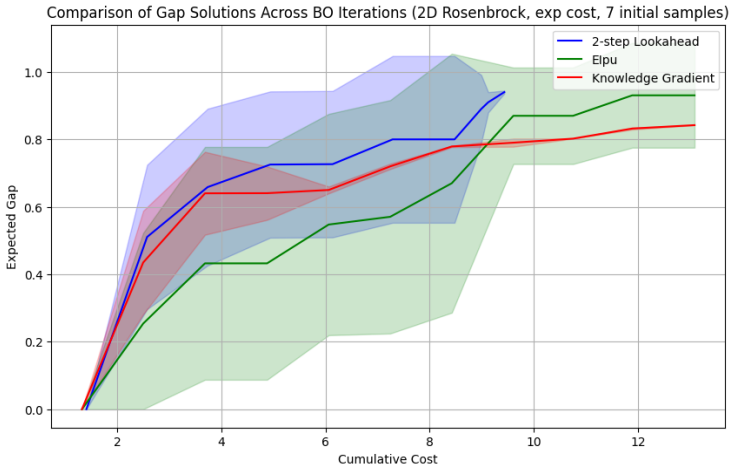

Figure 5: 2D Branin with 7 initial seeds and exponential cost

No methods achieve a gap of 1.0 here within the computational constraints, but we can see our proposed method has a steeper curve, so it finds better samples while accumulating less cost.

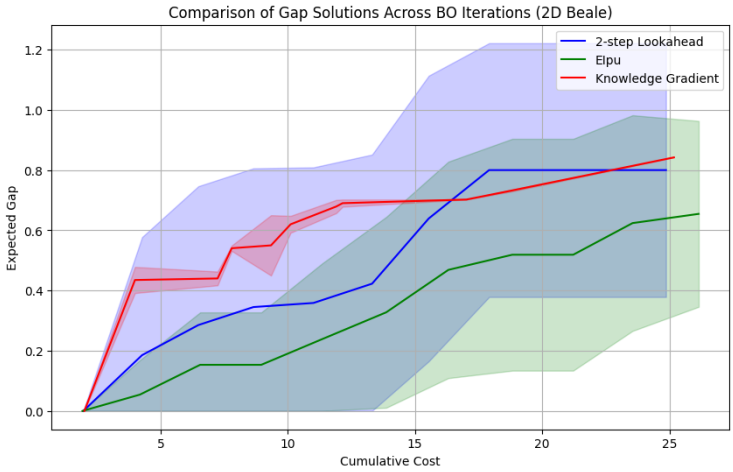

Figure 6: 2D Beale with 10 initial seeds and trigonometric cost

Knowledge gradient outperforms our method here, but 2-step look-ahead clearly outperforms $EIpu$.

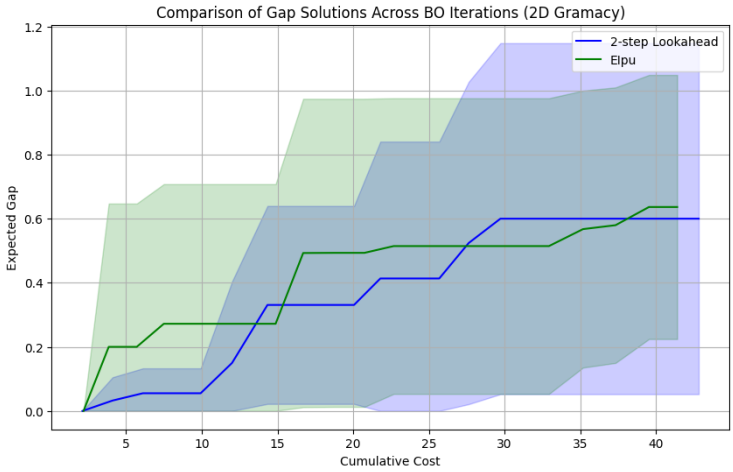

Figure 7: 2D Gramacy with 10 initial seeds and trigonometric cost

We only test against $EIpu$ here because our Google Colab environment crashed when running knowledge gradient with these synthetic functions.

A.7.3 Cost Accumulation Plots

These plots show the size of each sample of 2-step lookahead compared to $EIpu$ as each algorithm continues. The z-axis is cut off but it is the cost of each sample. The region to the left of each graph is where cumulative cost and expected gap is low, so this is the beginning of the algorithm. We can see the line is low on the z-axis here, so samples have low cost early on. The region to the right where the line is high on the z-axis is where gap is high and accumulated cost is high: this is toward the end of the algorithm. So each algorithm tends to sample low-cost regions first and high-cost regions later.

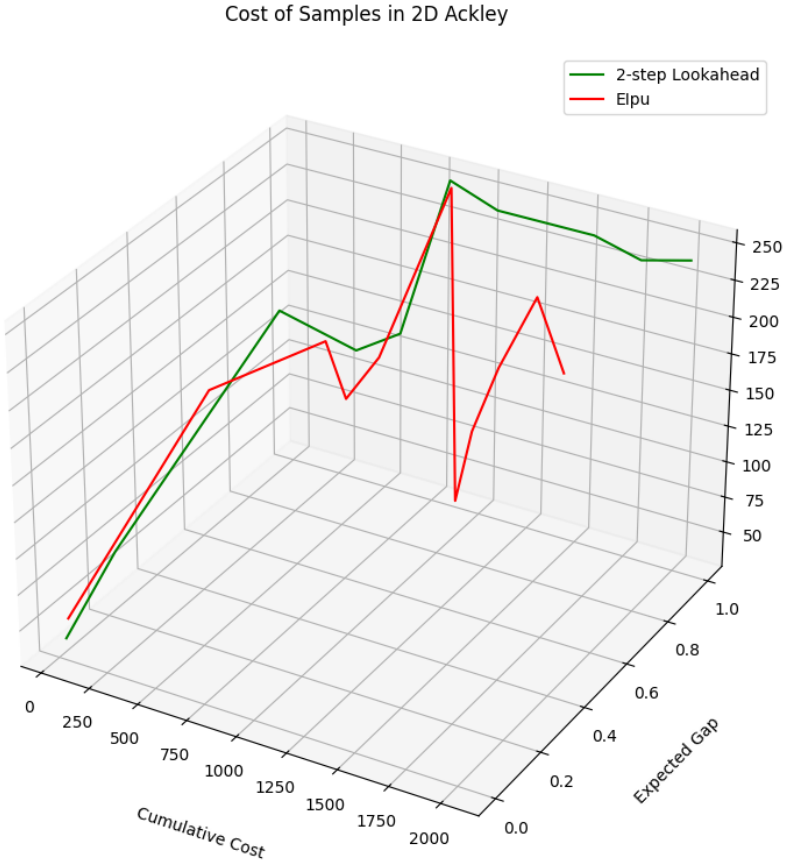

Figure 8: 2D Ackley cost accumulation

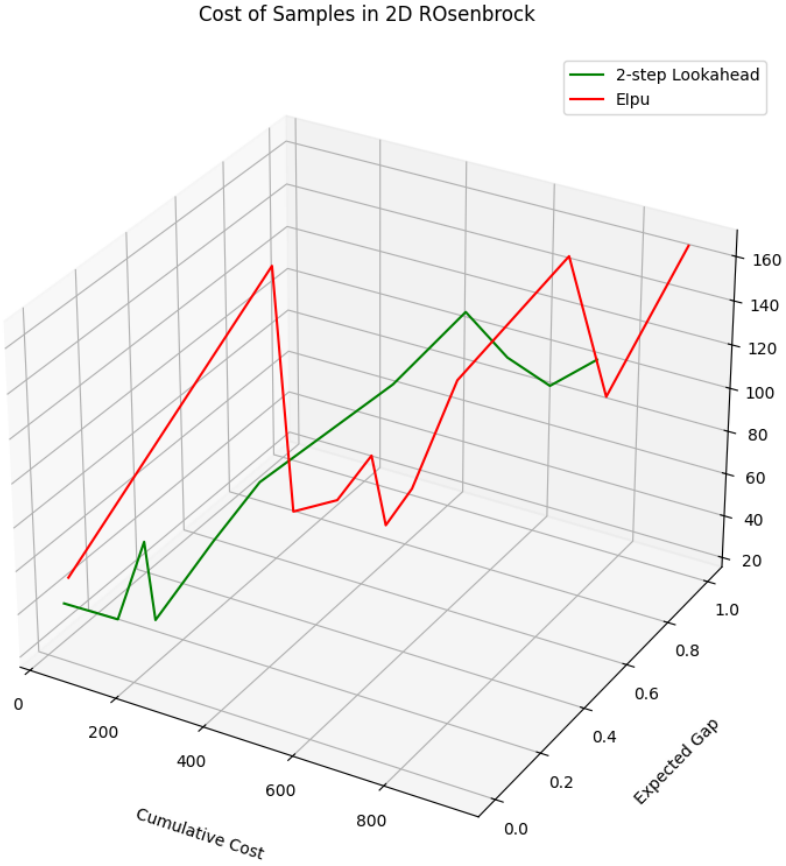

Figure 9: 2D Rosenbrock cost accumulation