Bayesian Optimization of Function Networks with Partial Evaluations

Poompol Buathong [*1] **Jiayue Wan** [*2] **Raul Astudillo** [3]
Samuel Daulton [4] **Maximilian Balandat** [4] **Peter I. Frazier** [2]

Abstract

Bayesian optimization is a powerful framework for optimizing functions that are expensive or time-consuming to evaluate. Recent work has considered Bayesian optimization of function networks (BOFN), where the objective function is given by a network of functions, each taking as input the output of previous nodes in the network as well as additional parameters. Leveraging this network structure has been shown to yield significant performance improvements. Existing BOFN algorithms for general-purpose networks evaluate the full network at each iteration. However, many real-world applications allow for evaluating nodes individually. To exploit this, we propose a novel knowledge gradient acquisition function that chooses which node and corresponding inputs to evaluate in a cost-aware manner, thereby reducing query costs by evaluating only on a part of the network at each step. We provide an efficient approach to optimizing our acquisition function and show that it outperforms existing BOFN methods and other benchmarks across several synthetic and real-world problems. Our acquisition function is the first to enable cost-aware optimization of a broad class of function networks.

1. Introduction

Bayesian optimization (BO) (Močkus, 1975; Frazier, 2018) has emerged as a powerful framework for optimizing functions with expensive or time-consuming evaluations. BO has proved its efficacy in a variety of applications, including hyperparameter tuning of machine learning models (Snoek et al., 2012), materials design (Frazier et al., 2008; Zhang et al., 2020), vaccine manufacturing (Rosa et al., 2022), and pharmaceutical product development (Sano et al., 2020).

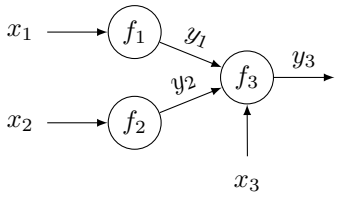

Figure 1: An example function network in the manufacturing problem.

In many applications, such as manufacturing (Ghasemi et al., 2018), epidemic model calibration (Garnett, 2002), machine learning pipeline optimization (Xin et al., 2021), and robotic control (Plappert et al., 2018), objective functions are computed by evaluating a network of functions where each function takes as input the outputs of its parent nodes. Consider the function network in Figure 1, which illustrates the stages of a manufacturing process. The process begins with a raw material described by x_1. This raw material is used to produce an intermediate part described by y_1 through a process f_1. Similarly, a second raw material described by x_2 is used to produce another intermediate part described by y_2 through a process f_2. These parts (with properties y_1 and y_2) are combined with another raw material described by x_3 in a process f_3 to make the final product, the quality of which is denoted by y_3. Our goal is to choose x_1, x_2, x_3 to maximize y_3.

Astudillo & Frazier (2021a) showed that utilizing intermediate outputs in the network, i.e., y_1 and y_2, to decide which design parameters $x = (x_1, x_2, x_3)$ to evaluate significantly improves the performance of BO. However, this and other prior work have not exploited the ability to perform *partial evaluations* of the function network, i.e., the ability to evaluate only a subset of nodes in the network at each iteration and use the so-obtained information to decide on the inputs to subsequent nodes, and potentially even to pause the evaluation process. As we demonstrate later, doing so can significantly improve performance, especially when evaluation costs vary significantly across nodes. For example, if evaluating f_1 is much cheaper than evaluating f_2, it may be

[*]Equal contribution [1]Center for Applied Mathematics, Cornell University [2]School of Operations Research and Information Engineering, Cornell University [3]Department of Computing and Mathematical Sciences, Caltech [4]Meta. Correspondence to: Poompol Buathong <pb482@cornell.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

advantageous to initially focus resources on understanding the range of values taken by y_1 before performing too many costly evaluations of f_2.

In this work, we introduce a BO algorithm that significantly improves performance over existing methods by taking advantage of the ability to perform partial evaluations. This algorithm iteratively selects a node in the function network and a corresponding input to evaluate it, with the goal of identifying the global optimum within a limited budget.

Our contributions are summarized as follows:

1. We introduce a framework for Bayesian optimization of function networks that allows partial evaluations.
2. We propose a knowledge-gradient-based acquisition function (p-KGFN) that, to our knowledge, is the first to actively leverage partial evaluations in general function networks in a cost-aware fashion.
3. We propose an approximation of p-KGFN that can be optimized more efficiently.
4. We demonstrate the benefits of exploiting partial evaluations through several numerical experiments, including both synthetic and real-world applications with a variety of network structures.

2. Related Work

Grey-box BO Our work falls within grey-box BO (Astudillo & Frazier, 2021b), which focuses on exploiting the known structure of the objective function (e.g., the function network structure considered in our work) to improve sampling decisions. BO of functions with a composite or network structure has been previously studied in the literature. For instance, Uhrenholt & Jensen (2019) considered objective functions that are sums of squared errors, while Astudillo & Frazier (2019) and Jain et al. (2023) considered a more general setting where the objective function is the composition of an expensive vector-valued inner function and a cheap outer function. BO of function networks was pioneered by Astudillo & Frazier (2021a), introducing a probabilistic model that exploits function network structure and pairing this model with the expected improvement (EI) acquisition function (Jones et al., 1998).

BO with Partial Evaluations The ability to perform partial evaluations in the context of BO of function networks has been studied for specific network structures. Kusakawa et al. (2022) considered function networks constituted by a chain of nodes and developed an algorithm that can pause an evaluation at an intermediate node. However, their approach, which uses an EI-based acquisition function, cannot be easily extended to quantify the value of evaluating a single node in more general function networks. Lin et al. (2021) explored a setting where changing values of a subset of variables corresponding to different stages in a pipeline incurs a "switching cost". Their approach assumes fully sequential dependence between stages and cannot reuse previous evaluations. Additionally, Lin et al. (2021) adopted a "slow-moving bandit" formulation that aims to minimize cumulative regret, whereas we seek to minimize simple regret. Outside the function networks setting, Hernández-Lobato et al. (2016) and Daulton et al. (2023) considered BO with partial evaluations for constrained and multi-objective optimization, respectively.

Cost-aware BO Our work is related to research considering heterogeneous evaluation costs across the search space. Our approach is similar in nature to those proposed by Snoek et al. (2012), Wu et al. (2020), and Daulton et al. (2023), whose acquisition functions value points based on the value of information per unit cost, thus favoring lower-cost evaluations. Lee et al. (2020) adopted a cost-cooling schedule that discourages high-cost points early in the BO loop, Abdolshah et al. (2019) incorporated cost-aware constraints while solving multi-objective BO problems, and Astudillo et al. (2021) and Lee et al. (2021) proposed non-myopic acquisition functions formulated using Markov decision processes for solving budgeted BO problems.

3. Problem Statement and Statistical Model

3.1. Problem Statement

Following the setup of Astudillo & Frazier (2021a), we consider a sequence of functions $f_1, f_2, \ldots, f_K$, arranged as nodes in a network representing the evaluation process. Specifically, the network structure is encoded as a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, K\}$ and $\mathcal{E} = \{(i, j) : f_j \text{ takes the output of } f_i \text{ as input}\}$ denote the sets of nodes and edges, respectively. We assume that the final node function, f_K, is scalar-valued. However, the other node functions may be vector-valued.

Let $\mathcal{J}(k)$ denote the parent nodes of node k. Without loss of generality, we assume that nodes are ordered such that $j < k$ for all $j \in \mathcal{J}(k)$. Let $\mathcal{I}(k) \subseteq \{1, 2, \ldots, d\}$ denote the set of components of the input vector $x \in \mathbb{X} \subset \mathbb{R}^d$ taken as an input by each function f_k.[1] The output of node k when the function network is evaluated at x is denoted by $y_k(x)$. The outputs $y_1(x), y_2(x), \ldots, y_K(x)$ can be computed recursively as

$$y_k(x) = f_k(y_{\mathcal{J}(k)}(x), x_{\mathcal{I}(k)}),\ k = 1, 2, \ldots, K. \quad (1)$$

For each node k, we assume that there is an associated

[1] This set may be empty for some nodes if they take as input only the outputs from their parent nodes.

known positive evaluation cost function $c_k(\cdot)$.[2] Our goal is to maximize the final node's function value $y_K(x)$ while minimizing the cumulative evaluation cost. To support this goal, our algorithm will select at each iteration a node k and corresponding input z_k at which f_k will be evaluated.

We distinguish two settings associated with the feasible values of z_k:

1. Evaluating a node k requires to previously obtain the outputs from its parent nodes, in which case z_k is comprised of the concatenation of these values and the additional parameters corresponding to node k.

2. The possible outputs of each node are known, and each node k can be evaluated at any feasible input (any admissible controllable input as well as any possible output of its parent nodes).

We focus on the first setting, which aligns with many practical situations. For example, in our manufacturing problem, executing a step requires the outputs of the preceding steps. Additionally, we restrict our attention to function networks where pairs of nodes do not share common inputs. This ensures there are valid combinations for evaluation at downstream nodes. However, this assumption can be relaxed by grouping nodes with shared inputs as a preprocessing step.

Finally, we consider the scenario in which each intermediate output is reusable. In other words, once a node's output is obtained, it can be repeatedly used in downstream evaluations. This scenario is common in settings such as machine learning (ML) pipeline optimization, where trained ML models can be saved and reused, or in large-batch manufacturing, where the manufactured batch volume is effectively infinite relative to the amounts required downstream.

3.2. Statistical Model

Following Astudillo & Frazier (2021a), we model the functions $f_1, f_2, \ldots, f_K$ as samples from independent Gaussian process (GP) prior distributions (Williams & Rasmussen, 2006). For each $k = 1, 2, \ldots, K$, let $\mu_{0,k}$ and $\Sigma_{0,k}$ denote the prior mean and covariance functions of f_k, respectively. Let $\mathcal{D}_{n,k} = \{(z_{j,k}, y_{j,k})\}_{j=1}^{n_k}$ denote the observations at node k after n iterations, where n_k is the number of observations at node k. The posterior distribution over f_k given $\mathcal{D}_{n,k}$ is a Gaussian process whose mean and covariance functions, denoted by $\mu_{n,k}$ and $\Sigma_{n,k}$, can be computed in closed form using the standard GP regression formulas (Williams & Rasmussen, 2006).

[2]When $c_k(\cdot)$ is unknown, we may learn it using a surrogate model and compute quantities involving costs by either taking the expectation over the distribution of $c_k(\cdot)$ or by replacing the cost function by the cost model's posterior mean.

Let $\mathcal{D}_n = \{\mathcal{D}_{n,k}\}_{k=1}^K$ denote the observations at all nodes after n iterations. The posterior distributions over $f_1, f_2, \ldots, f_K$ given $\mathcal{D}_n$ induce a posterior distribution on the final node value y_K. Although this distribution is generally non-Gaussian, we can obtain samples from it efficiently, as discussed in Section 5.2.

Our acquisition function, formally defined in Section 4, is constructed based on these posterior distributions and evaluation costs $c_k(\cdot)$. It quantifies the cost-normalized benefit of performing one additional partial evaluation at a specific node. Our BO algorithm then decides to evaluate at a node k^* with input $z_{k^*}^*$ yielding the maximum value of this acquisition function.

4. The p-KGFN Acquisition Function

Throughout this section, we assume that n samples have already been observed and are determining how to allocate sample $n+1$.

Let $\nu_n(x)$ denote the posterior mean of $y_K(x)$ given $\mathcal{D}_n$. Assuming risk-neutrality, the solution we would select if we were to stop at time n would be an x that maximizes the posterior mean of the final node's value,[3] i.e., a solution of

$$\nu_n^* = \max_{x \in \mathbb{X}} \nu_n(x). \tag{2}$$

Now, suppose one additional evaluation at a single node is allowed. For a node k with a given input z_k, observing $f_k(z_k)$ would result in an updated posterior over f_k, which in turn yields an updated posterior mean function of the final node value $\nu_{n+1}(\cdot)$ and also an updated maximum of the final node's posterior mean ν_{n+1}^*. The difference between the two quantities, i.e., $\nu_{n+1}^* - \nu_n^*$, quantifies the increment in the expected solution quality.

We note that $\nu_{n+1}^* - \nu_n^*$ is random at time n due to its dependence on the yet unobserved value of $f_k(z_k)$. Our acquisition function is obtained by taking the expectation of this increment with respect to the posterior on $f_k(z_k)$ and dividing it by the evaluation cost $c_k(z_k)$. Specifically, we define the knowledge gradient for function networks with partial evaluations (p-KGFN) by

$$\alpha_{n,k}(z_k) = \frac{\mathbb{E}_{y_k}[\nu_{n+1}^*] - \nu_n^*}{c_k(z_k)}. \tag{3}$$

The feasible set for z_k is given by $\mathbb{Z}_{n,k} := \mathbb{Y}_{n,\mathcal{J}(k)} \times \mathbb{X}_{\mathcal{I}(k)}$, where $\mathbb{Y}_{n,\mathcal{J}(k)}$ is the discrete set constituted by the outputs from the parent nodes of node k that have been previously generated after n iterations and $\mathbb{X}_{\mathcal{I}(k)}$ is the set of possible additional parameters at node k. Thus, at each iteration, the

[3]Note that we are concerned about the cost of evaluating a configuration *during but not after* the optimization.

next node and corresponding inputs to evaluate are given by

$$(k^*, z_{k^*}^*) \in \underset{k\in\{1,\dots,K\},\ z_k\in\mathbb{Z}_{n,k}}{\arg\max} \alpha_{n,k}(z_k). \tag{4}$$

Our acquisition function generalizes the classical knowledge gradient for regular BO (Frazier et al., 2008; Wu & Frazier, 2016). Moreover, it is cost-ware (in that it favors lower-cost evaluations at the same expected quality) and thus is similar in nature to the acquisition functions proposed by Snoek et al. (2012), Wu et al. (2020), and Daulton et al. (2023).

4.1. Advantages of Partial Evaluations

In this section, we illustrate the benefits of performing partial evaluations, as enabled by p-KGFN, through a simple two-stage function network example. Consider $f_1(x) = \sin(x) + 2\sin(2x)$ with domain $x \in [-4, 4]$, and $f_2(y) = \sin(3(y-1)/4)$, which takes as input the output of f_1. Additionally, assume that evaluation costs are constant given by $c_1 = 1$ for the first stage and $c_2 = 49$ for the second stage. We analyze the behavior of our proposed acquisition function, p-KGFN, and the acquisition function proposed by Astudillo & Frazier (2021a), EIFN, which also leverages the function network structure of the objective but requires full network evaluations at each iteration.

As shown in Figure 2, both EIFN and p-KGFN begin with three initial observations (black stars), evaluated across the full network. The initial models for $f_1(\cdot)$, $f_2(\cdot)$ and $f_2(f_1(\cdot))$ are presented in the first row. Both algorithms are allocated an evaluation budget of 150, which is equivalent to performing three evaluations of the full network. Rows two and three show the evaluations and resulting models upon budget depletion using EIFN and p-KGFN, respectively. We observe that EIFN makes decisions aimed at identifying the global maximum using the composite network model (third column) without realizing that the first function node is more complicated and that its evaluation is more cost-effective. Therefore, EIFN first chooses to evaluate in a region close to the initial inferred best solution (black square) and then performs two full evaluations, exploring areas with high uncertainty, such as the boundary at $x = 4$, and its inferred best solution upon budget depletion (purple square).

In contrast, p-KGFN takes evaluation costs into account and allocates the budget more efficiently. It first gathers information about the first function node through multiple evaluations (light green triangles) and then evaluates the second node only at the points that it considers most likely to improve the expected solution quality. This behavior yields a more efficient sampling policy which, in turn, results in a more accurate composite function model and inferred best solution (red square).

Similar behaviors emerge when comparing p-KGFN against KGFN with full evaluations, a knowledge-gradient-based acquisition function that also leverages function network but requires full evaluations (see Appendix G).

5. Maximization of p-KGFN

For simplicity, here we assume that $f_1, f_2, \dots, f_K$ are scalar-valued.[4] To solve (4), it suffices to solve

$$z_k^* \in \underset{z_k\in\mathbb{Z}_{n,k}}{\arg\max}\, \alpha_{n,k}(z_k) \tag{5}$$

for each node k. Recall that $\mathbb{Z}_{n,k} = \mathbb{Y}_{n,\mathcal{J}(k)} \times \mathbb{X}_{\mathcal{I}(k)}$, where $\mathbb{Y}_{n,\mathcal{J}(k)}$ is the discrete set constituted by the outputs from the parent nodes of node k that have been previously generated after n iterations. The discrete nature of $\mathbb{Y}_{n,\mathcal{J}(k)}$ makes solving (5) challenging. Additionally, solving (5) presents challenges due to the presence of nested expectations that cannot be computed in closed form, as we explain below.

To overcome the aforementioned challenges, we propose an approach to compute an approximate solution to (5). Our approach employs sample average approximation (SAA) (Kim et al., 2015; Balandat et al., 2020), which substitutes $\alpha_{n,k}(z_k)$ in (5) with a Monte Carlo (MC) estimate that is deterministic given a set of finite number of random variables independent of z_k. This is similar to the approach adopted by Astudillo & Frazier (2021a). Additionally, to further accelerate computation, we approximate ν_{n+1}^* by maximizing ν_{n+1} over a discretization of $\mathbb{X}$, which is similar to the approaches pursued by Scott et al. (2011) and Cakmak et al. (2020). Pseudo-code summarizing the approximate maximization of p-KGFN can be found in Appendix A.

5.1. Monte Carlo Estimation of the Outer Expectation

Recall that the outer expectation in the definition of p-KGFN is over the observation $y_k = f_k(z_k)$ that results from observing node k at z_k. To compute this expectation, we use the *reparametrization trick* (Kingma & Welling, 2013; Wilson et al., 2018) to generate samples from the posterior distribution on $f_k(z_k)$. Following Balandat et al. (2020), we call these *fantasy* samples. They are given by

$$\hat{y}_k^{(i)} = \mu_{n,k}(z_k) + \sigma_{n,k}(z_k)U^{(i)}$$

where $U^{(i)},\ i = 1, 2, \dots, I$ are i.i.d. standard normal random variables and $\mu_{n,k}(\cdot)$ and $\sigma_{n,k}(\cdot)$ denote the mean and standard deviation of the GP for node k at iteration n.

Each fantasy sample, were it actually observed, would generate a new posterior distribution. Let $\nu_{n+1}^{(i)}(x; z_k)$ denote the new posterior mean of y_K at x, conditioned on having observed $\hat{y}_k^{(i)}$. An unbiased estimator of $\alpha_{n,k}(z_k)$ is then

[4] Our framework can directly handle multi-output function nodes by employing a multi-output GP model (Alvarez et al., 2012) for each f_k.

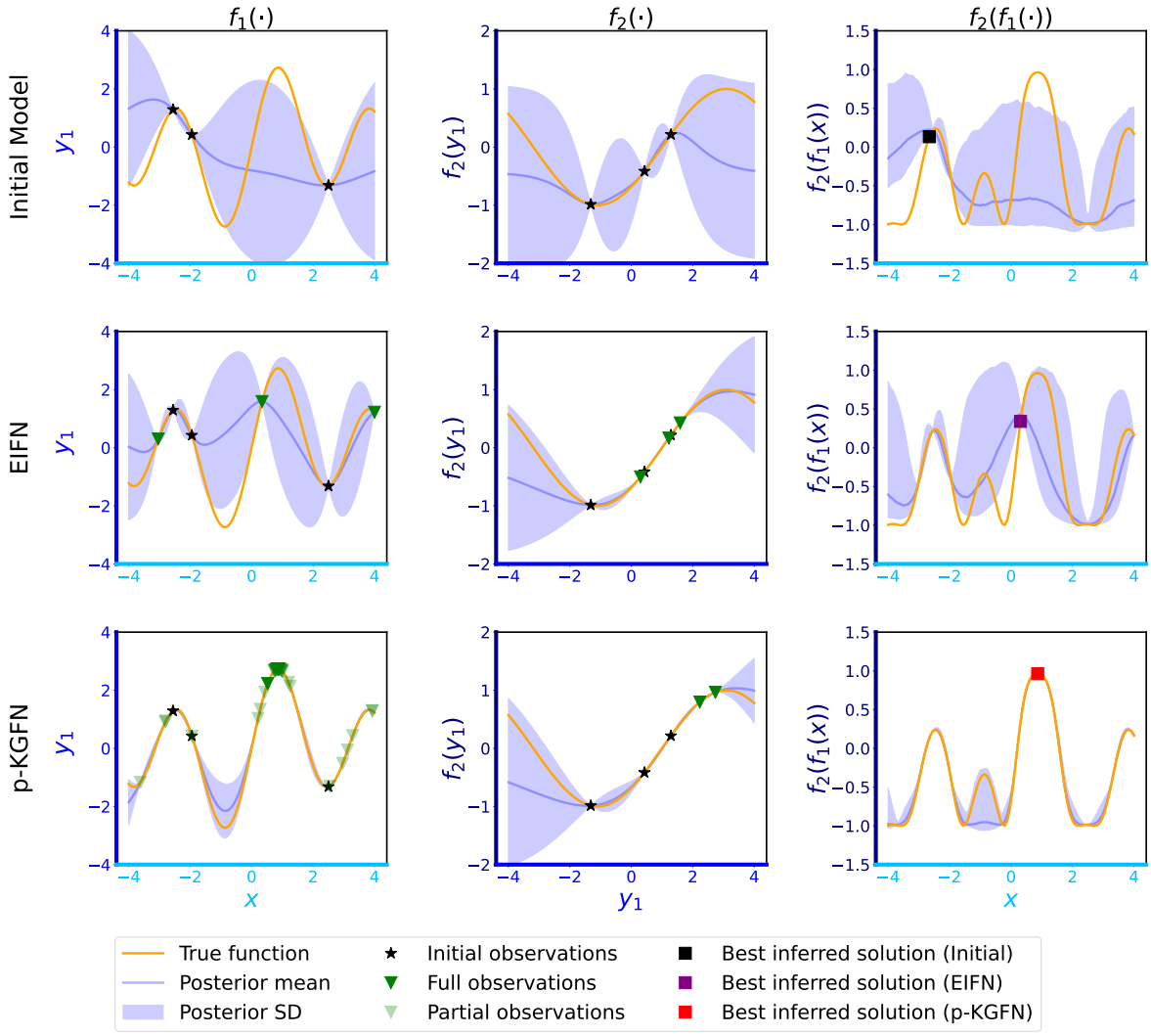

Figure 2: Comparison of EIFN and p-KGFN on a 1-D synthetic two-stage function network $f_2(f_1(\cdot))$. Top row (left to right): Initial models for $f_1(\cdot)$, $f_2(\cdot)$ and $f_2(f_1(\cdot))$. Second and third rows: Resulting surrogate models upon budget depletion by EIFN and p-KGFN, respectively. Each ground truth function is represented by an orange curve, while blue curves and shaded blue areas denote posterior mean and standard deviation, respectively. Black stars indicate the initial three points fully evaluated across the network for both algorithms. Dark green triangles represent the locations of full network evaluations. Light green triangles represent partial observations where only the first node was evaluated by p-KGFN. Black, purple, and red squares correspond to the initial and final inferred best solutions identified by EIFN and p-KGFN, respectively. We use the different colors for each axis to represent different types of inputs and outputs of the network as follows: light blue denotes the original input x to the network, dark blue denotes the output of the first node y_1, and dark navy denotes the output of the second node y_2.

given by

$$\left[\frac{1}{I}\sum_{i=1}^{I}\max_{x}\nu_{n+1}^{(i)}(x;z_k)-\nu_n^*\right]/\,c_k(z_k).$$

5.2. Monte Carlo Estimation of ν_{n+1}

We now discuss computation of $\nu_{n+1}^{(i)}(x;z_k)$. To explain our approach, we first describe how to generate a sample of the objective function value $y_K(x)$ under a particular posterior distribution. This approach is general, but we focus specifically on the posterior that defines $\nu_{n+1}^{(i)}(x;z_k)$. This is the posterior distribution that conditions on n previous observations and a new observation $\hat{y}_k^{(i)}$ of f_k at z_k. We refer to this distribution as the *fantasy-i posterior*.

Fix an index j and let $W^{(j)}=(W_1^{(j)},W_2^{(j)},\ldots,W_K^{(j)})^T\sim$

$\mathcal{N}(0, I_K)$. For a generic input x and the proposed point to sample z_k, define recursively over $\ell = 1, 2, \ldots, K$,

$$\begin{aligned}\hat{z}_\ell^{(i,j)}(x; z_k) &:= (\hat{y}_{\mathcal{J}(\ell)}^{(i,j)}(x; z_k), x_{I(\ell)}) \\ \hat{y}_\ell^{(i,j)}(x; z_k) &= \mu_{n+1,\ell}^{(i)}(\hat{z}_\ell^{(i,j)}(x; z_k)) \\ &\quad + \sigma_{n+1,\ell}^{(i)}(\hat{z}_\ell^{(i,j)}(x; z_k)) W_\ell^{(j)},\end{aligned} \tag{6}$$

where $\mu_{n+1,\ell}^{(i)}(\cdot)$ and $\sigma_{n+1,\ell}^{(i)}(\cdot)$ are the mean and standard deviation of the GP for node ℓ under the fantasy-i posterior. We use the notation $\hat{z}_\ell^{(i,j)}(\cdot; z_k)$ and $\hat{y}_\ell^{(i,j)}(\cdot; z_k)$ to indicate dependence of these quantities on $U^{(i)}$, $W^{(j)}$, and z_k.

By construction, $\hat{y}_K^{(i,j)}$ is a sample from the fantasy-i posterior over $y_K(x)$. Thus, we can approximate $\nu_{n+1}^{(i)}(x; z_k)$ by drawing many samples independently from a K-dimensional standard normal distribution and averaging the resulting final node value samples obtained via (6). For J samples, this estimate is given by $\frac{1}{J}\sum_{j=1}^{J} \hat{y}_K^{(i,j)}(x; z_k)$.

5.3. Putting the Pieces Together

We can now derive the following MC estimator of the p-KGFN acquisition function:

$$\hat{\alpha}_{n,k}(z_k) = \frac{\frac{1}{I}\sum_{i=1}^{I} \max_{x \in \mathbb{X}} \frac{1}{J}\sum_{j=1}^{J} \hat{y}_K^{(i,j)}(x; z_k) - \nu_n^*}{c_k(z_k)}. \tag{7}$$

We emphasize that the SAA approach relies on fixing the samples $U^{(i)},\ i = 1, 2, \ldots, I$, and $W^{(j)},\ j = 1, 2, \ldots, K$ that drive the above MC approximation as opposed to generating new samples for each x. Thus, the maximization of $\hat{\alpha}_{n,k}(z_k)$ can be seen as a deterministic optimization problem, and its solution as an estimator of the maximizer of $\alpha_{n,k}(z_k)$ (defined in (3)). Theorem 1 shows that this estimator does indeed converge to a maximizer of $\alpha_{n,k}(z_k)$ almost surely as the number of samples increases to infinity. We note that this result requires J to depend on I, so we write $J(I)$ to make this dependence explicit. The proof of Theorem 1 can be found in Appendix B.

Theorem 1. *Assume that the prior means $\mu_{0,k'}(\cdot)$ and variances $\sigma_{0,k'}(\cdot)$ are continuous and bounded for all nodes k', that $\mathbb{X}$ and $\mathbb{Z}_{n,k'}$ are compact, and that $\inf_{z \in \mathbb{Z}_{n,k'}} c_{k'}(z) > 0$, for all k'. Consider any node k and write $\hat{\alpha}_{n,k}(z)$ as $\hat{\alpha}_{n,k,I,J(I)}(z)$ to make the dependence on I and J explicit. Then, $\hat{\varphi}_{I,J(I)} := \max_{z \in \mathbb{Z}_{n,k}} \hat{\alpha}_{n,k,I,J(I)}(z)$ converges to $\varphi^* := \max_{z \in \mathbb{Z}_{n,k}} \alpha_{n,k}(z)$ almost surely as $I \to \infty$ where J is a function of I such that $\lim_{I \to \infty} J(I) = \infty$. Moreover, let $\hat{z}_{I,J(I)} \in \arg\max_{z \in \mathbb{Z}_{n,k}} \hat{\alpha}_{n,k,I,J(I)}(z)$ and $Z^* = \arg\max_{z \in \mathbb{Z}_{n,k}} \alpha_{n,k}(z)$. Then, the distance between $\hat{z}_{I,J(I)}$ and Z^* converges to zero almost surely as $I \to \infty$.*

5.4. Discretization of the Inner Problem

To speed up the maximization of $\hat{\alpha}_{n,k}(z_k)$, we discretize the set over which we take the maximum for each fantasy-i posterior in (7). I.e., rather than solving the inner maximization problem in (7) over the continuous domain $\mathbb{X}$, we instead solve it over a discrete set $\mathbb{A}$. Similar discretization approaches have been proposed in the literature (Scott et al., 2011; Ungredda et al., 2022).

The set $\mathbb{A}$ can be chosen through several heuristic approaches. Here, we choose $\mathbb{A}$ by taking into account two goals: exploring the promising domain based on the current statistical model, and exploiting the location of the current inferred best solution x_n^*. Hence, in each iteration, we form $\mathbb{A}$ using candidates generated by combining the following approaches: First, we draw N_T realizations from the posterior on y_K and include the maximizers of these realizations in $\mathbb{A}$. Second, we randomly generate N_L local points around x_n^*. We define a local point $x \in \mathbb{X}$ to be one for which $d(x, x_n^*) \leq r \max_{i=1,2,\ldots,d}(b_i - a_i)$, where a_i and b_i are the lower and upper bounds of i^{th} dimension input, respectively, and r is a positive hyperparameter. Finally, we also include the point x_n^* itself in $\mathbb{A}$.

An alternative approach to the discretization-based approach described above is to optimize $\hat{\alpha}_{n,k}(z_k)$ in a "one-shot" fashion (Balandat et al., 2020) by introducing a *fantasy variable* $x^{(i)}$ for each index i and then maximizing

$$\frac{\frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J} \hat{y}_K^{(i,j)}\left(x^{(i)}; z_k\right) - \nu_n^*}{c_k(z_k)}. \tag{8}$$

However, this approach results in an optimization problem where the dimension grows linearly in I, which in turn results in a substantial increase in computation time. In Appendix C.2, we compare one-shot optimization and the discretization-based approach we propose below in terms of compute time and solution quality.

5.5. Further Details

We maximize $\hat{\alpha}_{n,k}(z_k)$ by enumerating all available (previously evaluated) $y_{\mathcal{J}(k)} \in \mathbb{Y}_{n,\mathcal{J}(k)}$ and maximize the acquisition function over $x_{\mathcal{I}(k)} \in \mathbb{X}_{\mathcal{I}(k)}$ for each $y_{\mathcal{J}(k)}$ using gradients with respect to $x_{\mathcal{I}(k)}$, which we compute using auto-differentiation. Since $\hat{\alpha}_{n,k}(z_k)$ is deterministic, we use (quasi-)higher-order gradient-based methods, which have been shown to be fast and effective acquisition function optimizers (Daulton et al., 2020). We emphasize that this approach is trivially parallelizable: each maximization problem $\max_{z_k} \hat{\alpha}_{n,k}(z_k)$ for each previously evaluated $y_{\mathcal{J}(k)}$ *and* for each k can be solved independently and in parallel. Hence, the (wall-)time complexity for optimizing p-KGFN for each previously evaluated $y_{\mathcal{J}(k)}$ *and* for each k is the same as solving for a single node k from a single starting point $y_{\mathcal{J}(k)}$, given enough parallel compute resources.

6. Numerical Experiments

We evaluate p-KGFN against several benchmarks, including three algorithms that do not leverage the objective's function network structure: a random sampling baseline (Random), standard versions of expected improvement (EI) and knowledge gradient (KG), and three algorithms that do leverage network structure but require evaluation of the full network: EIFN (Astudillo & Frazier, 2021a), a slight modification of EIFN that uses the knowledge gradient instead of EI (KGFN),[5] and Thompson sampling for function networks (TSFN). While both p-KGFN and KGFN are one-step lookahead policies, KGFN considers full function network evaluations, whereas p-KGFN obtains one additional observation at one specific node. TSFN represents a simple acquisition function leveraging network structures constructed by a series of GP realizations sampled from the nodes' posterior distributions. All algorithms were implemented in BoTorch (Balandat et al., 2020). The code to reproduce our experiments is available at `https://github.com/frazier-lab/partial_kgfn`.

We assess performance on several function network structures, including single sequential networks, a multi-process network, and a multi-output network. Specifically, we explore two synthetic functions inspired by typical networks in materials design and manufacturing operations, as well as two real-world applications. In our experiments, we consider problems where the upstream nodes must be evaluated before downstream nodes. In this setting, exploiting partial evaluations is usually beneficial when the upstream node is cheaper to evaluate than the downstream node. When the situation is reversed, there are limited gains from using partial evaluations since the expensive upstream nodes must be evaluated before each evaluation of the cheaper downstream nodes. Motivated by real-world scenarios, here we focus on the case where initial nodes are cheaper than later nodes. Moreover, we consider problems without the upstream restriction in Appendix D.5.

In all experiments, each algorithm begins with $2d+1$ points chosen at random over the input space $\mathbb{X} \subseteq \mathbb{R}^d$. Each point $x \in \mathbb{X}$ is fully evaluated across the entire network (i.e., we observe $y_k(x)$ for $k = 1, \ldots, K$). Then, at each iteration, each algorithm sequentially selects a point to evaluate. All six baselines choose a point $x \in \mathbb{X}$ and evaluate the entire network. In contrast, p-KGFN can take advantage of partial evaluations by selecting both a node k and its input $z_k \in \mathbb{Z}_{n,k}$ to evaluate at each iteration. All experiments discussed in this section are noise-free. We conduct an additional experiment with noisy observations in Appendix D.6 to show the robustness of p-KGFN.

[5] KGFN has not been previously proposed in the literature. Our work is thus the first to describe KG policies for function networks with both partial and full evaluations.

We evaluate the performance of each algorithm by reporting at each iteration the ground truth value of $y_K(x_n^*)$, where $x_n^* \in \arg\max_{x \in \mathbb{X}} \nu_n(x)$. To highlight the benefits of partial evaluations, we utilize a posterior distribution for the final node value y_K obtained from a statistical model that incorporates the network structure discussed in Section 3.2 to compute the metric for all algorithms. Note that this favors algorithms such as EI, KG, and Random, which make decisions without leveraging the network structure (results when using a model not incorporating network structure are presented in Appendix H). Averaging over 30 replications, we report the mean of this metric with the error bars showing two standard errors.

6.1. Synthetic Test Problems

Ackley6D (Ackley) This problem is structured as a two-stage function network (Figure 3a), where the first stage takes a 6-dimensional input and the second stage takes as an input the output of the first stage. The node function f_1 is the negated Ackley function (Jamil & Yang, 2013), and f_2 is given by $f_2(y_1) = -y_1 \times \sin(5y_1/6\pi)$. This network structure is commonly found in materials design, sequential processes, and multi-fidelity settings. In many applications, the early stages in a path through the function network are cheaper to evaluate than subsequent stages. For example, the first node can be an approximation or partial evaluation of a subsequent node. We therefore assume the costs are given by $c_1 = 1$ and $c_2 = 49$.

Manufacturing Network (Manu-GP) Motivated by the manufacturing application discussed in Section 1, we build a two-process network where the outputs of the two processes are combined at a final node (Figure 3b). The first process has two sequential nodes and its initial input is 2-dimensional. The second process has one node which takes a different 2-dimensional input. This network structure is typical in scenarios where individual components are produced through independent processes and combined to create a final product (e.g., chemical synthesis processes). In this experiment, we employ a sample path drawn from a GP prior for each function node. This is intended to emulate the variations in the characteristics of intermediate/final products, with respect to different design parameters. Since different processes typically have different levels of complexity, resulting in heterogeneous costs, we assume that $c_1 = 5$ and $c_2 = 10$ in the first process, and $c_3 = 10$ in the second process. As the final stage usually involves both component assembly and product quality assessment, we assume the final stage incurs a relatively high cost, $c_4 = 45$.

6.2. Real-World Applications

Molecular Design (FreeSolv) We consider the FreeSolv dataset (Mobley & Guthrie, 2014), which consists of cal-

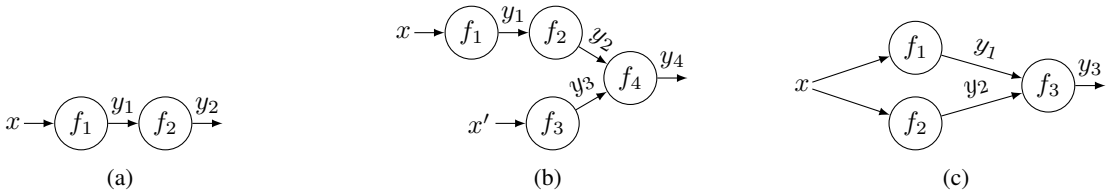

Figure 3: Function network structures in the numerical experiments: (a) Ackley and FreeSolv, (b) Manu-GP, and (c) Pharma.

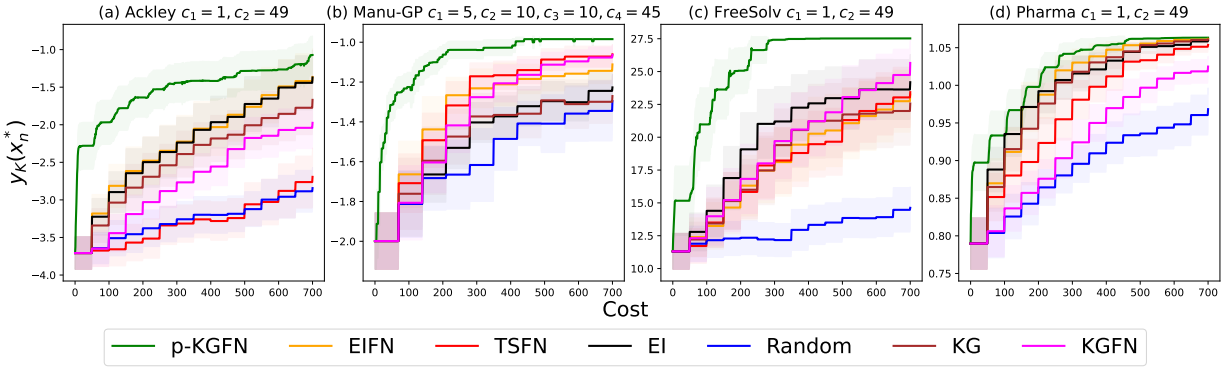

Figure 4: Optimization performance comparing between our proposed p-KGFN and benchmarks including EIFN, KGFN, TSFN, EI, KG and Random on four experiments: (a) Ackley, (b) Manu-GP, (c) FreeSolv, and (d) Pharma.

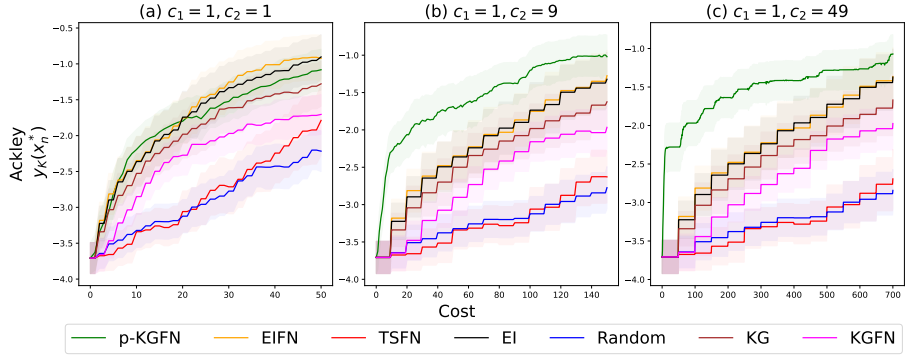

Figure 5: Cost sensitivity analysis for Ackley problem with different costs (a) $c_1 = 1, c_2 = 1$; (b) $c_1 = 1, c_2 = 9$; and (c) $c_1 = 1, c_2 = 49$.

culated and experimental hydration-free energies of 642 small molecules. A continuous representation is derived from the SMILES representation of each molecule through a variational autoencoder model (Gómez-Bombarelli et al., 2018). We apply principal component analysis (PCA) to reduce the dimension of these representations to three. In the context of materials design, our objective is to minimize the experimental free energy. We formulate this problem as a two-stage function network (Figure 3a). The first node takes the 3-dimensional representation of molecules as input and outputs the negative calculated free energy. The second node takes this output as input and returns the negative experimental free energy, which is our target for maximization. We fit GP surrogate models for both nodes based on the entire available dataset, which allows us to consider a continuous optimization domain. As in the Ackley problem, we assume $c_1 = 1$ and $c_2 = 49$.

Pharmaceutical Product Development (Pharma) An orally disintegrating tablet (ODT) is a drug dosage form designed to dissolve on the tongue. To ensure the production of high-quality ODTs, one must consider two crucial properties: disintegration time (f_1) and tensile strength (f_2). We employ surrogate models proposed in Sano et al. (2020) for these two target properties as functions of four input variables in the production process. In the same study, a simple deterministic score function (f_3), which combines

these two targets, is proposed to measure the ODT quality (Figure 3c). Our goal is to find the four input variables that maximize the score produced by node f_3. While both target properties are equally significant in determining the quality of the ODT, their measurements typically involve different levels of complexity. To reflect this, costs are set at $c_1 = 1$ for f_1 and $c_2 = 49$ for f_2. The f_3 score function, being already known and cost-free to evaluate, is not included as a node in the p-KGFN optimization process.

6.3. Results and Discussion

Figure 4 shows the performance of p-KGFN compared to various baselines across different experiments. While the baselines require full evaluations of the function networks, resulting in regular-spaced steps in their performance curves, p-KGFN operates without this limitation. We note that p-KGFN outperforms all baselines across all experiments.

In the FreeSolv and Ackley experiments, similar to the toy example from Section 4.1, p-KGFN demonstrates strategic budget allocation by learning the first function node thoroughly before deciding to explore the second node, and only in regions with high potential values. To assess the impact of evaluation costs on performance, we conduct a sensitivity analysis where we vary the cost of evaluating the second node in two-node network problems, as shown in Figure 5 and detailed in Appendix E. We observe that when the costs of the two nodes are equal, p-KGFN typically performs full evaluations, achieving performance levels comparable to baseline algorithms. The advantages of partial evaluations become more pronounced as the evaluation cost of the second node increases.

Comparing results across different test problems, we note that p-KGFN achieves more significant performance gains in scenarios where downstream nodes in the function network show strong correlations with their parent nodes (as seen in the Ackley and FreeSolv experiments) and involve higher evaluation costs. Such dynamics are common in real-world settings, where an upstream node might simulate a scenario that is physically tested in a downstream node. To explore the robustness of p-KGFN, we conduct additional experiments where upstream nodes are more challenging to model than downstream nodes, detailed in Appendix D.7. In such experiments, p-KGFN's performance remains on par with other benchmarks, confirming its effectiveness across diverse problem settings.

Overall, the networks evaluated in the numerical experiments showcase p-KGFN's capability to effectively manage a diverse range of network structures. These include sequential networks, multi-process networks, and multi-output networks with clearly defined objectives.

7. Conclusion

In this work, we considered Bayesian optimization of objectives represented by a network of functions, where individual nodes in the network can be evaluated independently. We proposed a new acquisition function for this class of problems, p-KGFN, that leverages the objective's function network structure along with the ability to evaluate individual nodes to improve sampling efficiency. Our numerical experiments on both synthetic and real-world problems demonstrate that our approach can significantly reduce evaluation costs and provide higher-quality solutions.

While our method offers substantial benefits through partial evaluations, it is also subject to some limitations. Specifically, optimizing the p-KGFN acquisition function requires considerable computational resources as it considers all nodes and available outputs at each iteration, which could be challenging for large networks (the average runtime for optimizing each acquisition function is reported in Appendix F). Nonetheless, in many real-world scenarios where evaluation costs are high, the savings achieved through an improved query strategy significantly outweigh the additional computational time. Additionally, it may be possible to extend and integrate the stock reduction technique from Kusakawa et al. (2022), designed for cascade-type networks, to a broader range of network structures to reduce the number of optimization problems considered in p-KGFN. We leave this as a direction for future work. Finally, our algorithm, like other knowledge-gradient-based algorithms, looks only a single step ahead. An interesting research direction would be to explore multi-step lookahead acquisition functions (Jiang et al., 2020) for function networks, though this would further increase the computational cost.

Acknowledgements

We would like to thank the anonymous reviewers for their comments. We also thank Eytan Bakshy for his valuable feedback. PB would like to thank DPST scholarship project granted by IPST, Ministry of Education, Thailand for providing financial support. PF and JW were supported by AFOSR FA9550-19-1-0283 and FA9550-20-1-0351.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abdolshah, M., Shilton, A., Rana, S., Gupta, S., and Venkatesh, S. Cost-aware multi-objective Bayesian opti-

misation. *arXiv preprint arXiv:1909.03600*, 2019.

Alvarez, M. A., Rosasco, L., Lawrence, N. D., et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Astudillo, R. and Frazier, P. Bayesian optimization of composite functions. In *International Conference on Machine Learning*, pp. 354–363. PMLR, 2019.

Astudillo, R. and Frazier, P. Bayesian optimization of function networks. *Advances in Neural Information Processing Systems*, 34:14463–14475, 2021a.

Astudillo, R. and Frazier, P. I. Thinking inside the box: A tutorial on grey-box Bayesian optimization. In *2021 Winter Simulation Conference (WSC)*, pp. 1–15. IEEE, 2021b.

Astudillo, R., Jiang, D., Balandat, M., Bakshy, E., and Frazier, P. Multi-step budgeted Bayesian optimization with unknown evaluation costs. *Advances in Neural Information Processing Systems*, 34:20197–20209, 2021.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, 33:21524–21538, 2020.

Cakmak, S., Astudillo Marban, R., Frazier, P., and Zhou, E. Bayesian optimization of risk measures. *Advances in Neural Information Processing Systems*, 33:20130–20141, 2020.

Daulton, S., Balandat, M., and Bakshy, E. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, 33:9851–9864, 2020.

Daulton, S., Balandat, M., and Bakshy, E. Hypervolume knowledge gradient: A lookahead approach for multi-objective Bayesian optimization with partial information. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 7167–7204, 2023.

Frazier, P. I. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Frazier, P. I., Powell, W. B., and Dayanik, S. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

Garnett, G. P. An introduction to mathematical models in sexually transmitted disease epidemiology. *Sexually Transmitted Infections*, 78(1):7–12, 2002.

Ghasemi, A., Heavey, C., and Laipple, G. A review of simulation-optimization methods with applications to semiconductor operational problems. In *2018 Winter Simulation Conference (WSC)*, pp. 3672–3683. IEEE, 2018.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Hernández-Lobato, J. M., Gelbart, M. A., Adams, R. P., Hoffman, M. W., and Ghahramani, Z. A general framework for constrained Bayesian optimization using information-based search. *Journal of Machine Learning Research*, 17(160):1–53, 2016.

Jain, K., Bodas, T., et al. Bayesian optimization for function compositions with applications to dynamic pricing. *arXiv preprint arXiv:2303.11954*, 2023.

Jamil, M. and Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

Jiang, S., Jiang, D., Balandat, M., Karrer, B., Gardner, J., and Garnett, R. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33:18039–18049, 2020.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

Kim, S., Pasupathy, R., and Henderson, S. G. A guide to sample average approximation. *Handbook of Simulation Optimization*, pp. 207–243, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kusakawa, S., Takeno, S., Inatsu, Y., Kutsukake, K., Iwazaki, S., Nakano, T., Ujihara, T., Karasuyama, M., and Takeuchi, I. Bayesian optimization for cascade-type multistage processes. *Neural Computation*, 34(12):2408–2431, 2022.

Lee, E. H., Perrone, V., Archambeau, C., and Seeger, M. Cost-aware Bayesian optimization. *arXiv preprint arXiv:2003.10870*, 2020.

Lee, E. H., Eriksson, D., Perrone, V., and Seeger, M. A nonmyopic approach to cost-constrained Bayesian optimization. In *Uncertainty in Artificial Intelligence*, pp. 568–577. PMLR, 2021.

Lin, C.-H., Miano, J. D., and Dyer, E. L. Bayesian optimization for modular black-box systems with switching costs. In *Uncertainty in Artificial Intelligence*, pp. 1024–1034. PMLR, 2021.

Mobley, D. L. and Guthrie, J. P. Freesolv: A database of experimental and calculated hydration free energies, with input files. *Journal of Computer-Aided Molecular Design*, 28:711–720, 2014.

Močkus, J. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference: Novosibirsk, July 1–7, 1974*, pp. 400–404. Springer, 1975.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Rosa, S. S., Nunes, D., Antunes, L., Prazeres, D. M., Marques, M. P., and Azevedo, A. M. Maximizing mRNA vaccine production with Bayesian optimization. *Biotechnology and Bioengineering*, 119(11):3127–3139, 2022.

Sano, S., Kadowaki, T., Tsuda, K., and Kimura, S. Application of Bayesian optimization for pharmaceutical product development. *Journal of Pharmaceutical Innovation*, 15: 333–343, 2020.

Scott, W., Frazier, P., and Powell, W. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026, 2011.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

Uhrenholt, A. K. and Jensen, B. S. Efficient Bayesian optimization for target vector estimation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2661–2670. PMLR, 2019.

Ungredda, J., Pearce, M., and Branke, J. Efficient computation of the knowledge gradient for Bayesian optimization. *arXiv preprint arXiv:2209.15367*, 2022.

Williams, C. K. and Rasmussen, C. E. *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA, 2006.

Wilson, J., Hutter, F., and Deisenroth, M. Maximizing acquisition functions for Bayesian optimization. *Advances in neural information processing systems*, 31, 2018.

Wu, J. and Frazier, P. The parallel knowledge gradient method for batch Bayesian optimization. *Advances in Neural Information Processing Systems*, 29, 2016.

Wu, J., Toscano-Palmerin, S., Frazier, P. I., and Wilson, A. G. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pp. 788–798. PMLR, 2020.

Xin, D., Miao, H., Parameswaran, A., and Polyzotis, N. Production machine learning pipelines: Empirical analysis and optimization opportunities. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 2639–2652, 2021.

Zhang, Y., Apley, D. W., and Chen, W. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific Reports*, 10(1):4924, 2020.

A. Pseudo-Code for the p-KGFN Algorithm

We present the pseudo-code for implementing Bayesian optimization with the p-KGFN acquisition function, supplementing the descriptions in Section 5. Algorithm 1 outlines the BO loop employing the p-KGFN algorithm. Algorithm 2 describes the computation of the MC estimate of the acquisition value. Algorithm 3 describes how we estimate the posterior mean of the final function node via MC simulation, which is necessary for Algroithm 2. We defer the discussion of acquisition optimization to Appendix C, where we compare our implementation, optimization-via-discretization, against a commonly used approach, one-shot optimization.

Algorithm 1 Bayesian Optimization using p-KGFN

Input:
$c_k(\cdot)$, the evaluation cost function for node k, $k = 1, \ldots, K$
B, the total evaluation budget
$\mu_{0,k}(\cdot)$ and $\sigma_{0,k}(\cdot)$, the mean and standard deviation of the GP for node k, $k = 1, \ldots, K$ (fitted using initial observations)
Output: the point with the largest posterior mean at the final function node

$n \leftarrow 0$
$b \leftarrow 0$
while $b < B$ **do**
  $n \leftarrow n + 1$
  for $k = 1, \ldots, K$ **do**
    identify the set of combinations of previously evaluated $y_{\mathcal{J}(k)}$, $\mathbb{Y}_{n,\mathcal{J}(k)}$
    if $\mathbb{Y}_{n,\mathcal{J}(k)} = \emptyset$ **then**
      $\hat{\alpha}^*_{n,k} \leftarrow -1$
    else
      $\hat{\alpha}^*_{n,k} \leftarrow \max_{z \in \mathbb{Z}_{n,k}} \hat{\alpha}_{n,k}(z)$ where $\hat{\alpha}_{n,k}(\cdot)$ is computed using Algorithm 2
      $z^*_k \leftarrow \arg\max_{z \in \mathbb{Z}_{n,k}} \hat{\alpha}_{n,k}(z)$
      if $c_k(z^*_k) > B - b$ **then**
        $\hat{\alpha}^*_{n,k} \leftarrow -1$
      end if
    end if
  end for
  if $\max_k \hat{\alpha}^*_{n,k} = -1$ **then**
    break
  else
    $k^* \leftarrow \arg\max_{k \in \{1,\ldots,K\}} \hat{\alpha}^*_{n,k}$
    obtain $y_{k^*} = f_{k^*}(z^*_{k^*})$
    update the GP model for node k^* with the additional observation $(z^*_{k^*}, y_{k^*})$
    $b \leftarrow b + c_{k^*}(z^*_{k^*})$
  end if
end while

return $\arg\max_{x \in \mathbb{X}} \hat{\nu}_n(x)$, an estimate of $\nu_n(x)$ given in Algorithm 3 using a gradient-based method

Algorithm 2 MC Estimate of $\alpha_{n,k}(z_k)$

Input:
k, the node to be evaluated
z_k, the input for node k
$c_k(\cdot)$, the evaluation cost function for node k
I, the number of fantasy observations to create
J, the number of MC samples for estimating the posterior mean of the final function node
$\mu_{n,k}(\cdot)$ and $\sigma_{n,k}(\cdot)$, the mean and standard deviation of the GP for node k, $k = 1, \ldots, K$
Output: $\hat{\alpha}_{n,k}(z_k)$, the estimated acquisition value

1: solve $\max_{x \in \mathbb{X}} \hat{\nu}_n(x)$, an estimate of $\nu_n(x)$ given in Algorithm 3 using a gradient-based method and obtain $\hat{\nu}_n^*$
2: generate I independent standard normal random variables $U^{(i)}$, $i = 1, \ldots, I$
3: **for** $i = 1, \ldots, I$ **do**
4: $\quad \hat{y}_k^{(i)} \leftarrow \mu_{n,k}(z_k) + \sigma_{n,k}(z_k) U^{(i)}$
5: $\quad$ update the posterior of GP for node k with the additional observation $\left(z_k, \hat{y}_k^{(i)}\right)$
6: $\quad$ solve $\max_{x \in \mathbb{X}} \hat{\nu}_{n+1}^{(i)}(x)$ to obtain $\hat{\nu}_{n+1}^{(i)*}$ for fantasy-i model using Algorithm 3
7: **end for**
8: $\hat{\alpha}_{n,k}(z_k) \leftarrow \frac{1}{c_k(z_k)} \left(\frac{1}{I} \sum_{i=1}^{I} \hat{\nu}_{n+1}^{(i)*} - \hat{\nu}_n^* \right)$
9: **return** $\hat{\alpha}_{n,k}(z_k)$

Note that in Line 6 of Algorithm 2, the samples $W^{(j)}$ used in Algorithm 3 are shared across the MC approximation for all fantasy-i models.

Algorithm 3 Posterior Mean Estimate of the Final Function Node via MC Simulation

Input:
$x \in \mathbb{X}$, a design point of the function network
J, the number of MC samples
$\mu_k(\cdot)$ and $\sigma_k(\cdot)$, the mean and standard deviation of the GP for node k, for $k = 1, \ldots, K$
Output: $\hat{\nu}(x)$, the estimated posterior mean

1: generate J independent samples $W^{(j)} = \left(W_1^{(j)}, W_2^{(j)}, \ldots, W_K^{(j)}\right)^T$ for $j = 1, \ldots, J$ from $\mathcal{N}(0, I_K)$;
2: **for** $j = 1, \ldots, J$ **do**
3: $\quad$ **for** $k = 1, \ldots, K$ **do**
4: $\quad\quad$ define $\hat{z}_k^{(j)}(x) := (\hat{y}_{\mathcal{J}(k)}^{(j)}(x), x_{\mathcal{I}(k)})$
5: $\quad\quad \hat{y}_k^{(j)}(x) \leftarrow \mu_k(\hat{z}_k^{(j)}(x)) + \sigma_k(\hat{z}_k^{(j)}(x)) W_k^{(j)}$
6: $\quad$ **end for**
7: **end for**
8: $\hat{\nu}(x) \leftarrow \frac{1}{J} \sum_{j=1}^{J} \hat{y}_K^{(j)}(x)$
9: **return** $\hat{\nu}(x)$

B. Proof of Theorem 1

In this section, we prove the following theorem.

Theorem 1. *Assume that the prior means $\mu_{0,k'}(\cdot)$ and variances $\sigma_{0,k'}(\cdot)$ are continuous and bounded for all nodes k', that $\mathbb{X}$ and $\mathbb{Z}_{n,k'}$ are compact, and that $\inf_{z \in \mathbb{Z}_{n,k'}} c_{k'}(z) > 0$, for all k'. Consider any node k and write $\hat{\alpha}_{n,k}(z)$ as $\hat{\alpha}_{n,k,I,J(I)}(z)$ to make the dependence on I and J explicit. Then, $\hat{\varphi}_{I,J(I)} := \max_{z \in \mathbb{Z}_{n,k}} \hat{\alpha}_{n,k,I,J(I)}(z)$ converges to $\varphi^* := \max_{z \in \mathbb{Z}_{n,k}} \alpha_{n,k}(z)$ almost surely as $I \to \infty$ where J is a function of I such that $\lim_{I \to \infty} J(I) = \infty$. Moreover, let $\hat{z}_{I,J(I)} \in \arg\max_{z \in \mathbb{Z}_{n,k}} \hat{\alpha}_{n,k,I,J(I)}(z)$ and $Z^* = \arg\max_{z \in \mathbb{Z}_{n,k}} \alpha_{n,k}(z)$. Then, the distance between $\hat{z}_{I,J(I)}$ and Z^* converges to zero almost surely as $I \to \infty$.*

Our proof is mainly based on Lemma A1 and Theorem A1 in Rubinstein & Shapiro (1993). We will assume throughout this section the assumptions in the statement of Theorem 1 — that the prior means $\mu_{0,k'}(\cdot)$ and variances $\sigma_{0,k'}(\cdot)$ are continuous

and bounded for all k', that $\mathbb{X}$ and $\mathbb{Z}_{n,k'}$ are compact, and that the cost $c_{k'}(z)$ is bounded below away from 0.

To support this proof, we first introduce the notation used in the statements and proofs of the lemmas below. Observe that the posterior distribution of f_k given the data up to time n and an additional observation of f_k at z is a Gaussian process whose posterior mean and posterior variance can be written using the standard Gaussian process regression formulas (Williams & Rasmussen, 2006):

$$\mu_{n+1,k}(z') = \mu_{n,k}(z') + \Sigma_{n,k}(z', z)(\Sigma_{n,k}(z, z) + \lambda)^{-1}(f_k(z) - y_{n,k}(z)),$$

and

$$\begin{aligned}\sigma_{n+1,k}(z') &= \Sigma_{n,k}(z', z') - \Sigma_{n,k}(z', z)(\Sigma_{n,k}(z, z) + \lambda)^{-1}\Sigma_{n,k}(z', z) \\ &= \Sigma_{n,k}(z', z') - \Sigma_{n,k}(z', z)^2/(\Sigma_{n,k}(z, z) + \lambda).\end{aligned}$$

Let $U = (f_k(z) - y_{n,k}(z))/\sqrt{\Sigma_{n,k}(z, z) + \lambda}$, which is a standard normal random variable. Then, we can rewrite $\mu_{n+1,k}$ as

$$\mu_{n+1,k}(z') = \mu_{n,k}(z') + \Sigma_{n,k}(z', z)U/\sqrt{\Sigma_{n,k}(z, z) + \lambda}.$$

Define $\mu_{n+1,\ell} = \mu_{n,\ell}$ and $\sigma_{n+1,\ell} = \sigma_{n,\ell}$ for $\ell \neq k$.

Now, we can create a recursive expression for a sample from the posterior over $y_K(x)$ given these $n + 1$ evaluations in terms of the posterior means and variances and a sequence of standard normal random variables $W = (W_\ell : \ell = 1, \ldots, K)$. Specifically, define recursively for each $\ell = 1, \ldots, K$,

$$\begin{aligned}z_\ell(x; z) &:= (y_{J(\ell)}(x; z), x_{I(\ell)}) \\ y_\ell(x; z) &= \mu_{n+1,\ell}(z_\ell(x; z)) + \sigma_{n+1,\ell}(z_\ell(x; z))W_\ell.\end{aligned}$$

Making a slight abuse, we make the dependence of $y_K(x)$ on, z, U, and W explicit through the notation $y_K(U, W, x, z)$.

Now, consider a specific node k. We drop the subscript n, k for convenience. We define $h(U, W, x, z) = \frac{y_K(U,W,x,z)}{c(z)}$. In addition, define

$$\alpha(z) = \mathbb{E}[\max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z)|U]]$$

and

$$\hat{\alpha}_{I,J(I)}(z) = \frac{1}{I}\sum_{i=1}^{I} \max_{x\in\mathbb{X}} \frac{1}{J(I)} \sum_{j=1}^{J(I)} h(U_i, W_j, x, z),$$

where $J(I)$ will be defined later.

Before proving the theorem 1, we prove several auxiliary lemmas.

Lemma 1. *For almost every (u, w), $h(u, w, \cdot, \cdot)$ is continuous.*

Proof of Lemma 1. Since the prior means $\mu_{0,k'}(\cdot)$ variances $\sigma_{0,k'}(\cdot)$ are continuous, a simple argument shows that the posterior means $\mu_{n+1,k'}(\cdot)$ and variances $\sigma_{n+1,k'}(\cdot)$ are continuous too. Thus, $h(u, w, x, z)$ is a composition of continuous functions and so is also continuous. □

Lemma 2. *There exist finite non-negative constants c_0, c_1 and c_2 not depending on U, W, x or z such that $|y_K(U, W, x, z)| \leq c_0 + c_1|W_K| + c_2|U|$.*

Proof of Lemma 2. Consider two cases. In the first case, suppose that the node we are measuring, k, precedes the final node, $k < K$. In this case, the posterior mean and variance of node K do not change and are equal to their values under the prior. Thus, $y_K(U, W, x, z)$ is equal to the prior mean function $\mu_{n,K}(\cdot)$ evaluated at a random input $z_K(x; z)$ plus the noise term $\sigma_{n,K}(z_K(x; z))W_K$. This is bounded above by $\sup_{z'\in\mathbb{Z}_{n,K}} |\mu_{n,K}(z')| + \sup_{z'\in\mathbb{Z}_{n,K}} |\sigma_{n,K}(z')||W_K|$. Note that $c_0 = \sup_{z'\in\mathbb{Z}_{n,K}} |\mu_{n,K}(z')|$ and $c_1 = \sup_{z'\in\mathbb{Z}_{n,K}} |\sigma_{n,K}(z')|$ are finite.

In the second case, we measure the final node, $k = K$. Then,

$$\mu_{n+1,K}(z') = \mu_{n,K}(z') + \tilde{\sigma}_{n,K}(z', z)U$$

where $\tilde{\sigma}_{n,K}(z', z) = \Sigma_{n,K}(z', z)/\sqrt{\Sigma_{n,K}(z, z) + \lambda}$.

Thus,

$$y_K(U, W, x, z) = \mu_{n,K}(z_K(x; z)) + \tilde{\sigma}_{n,K}(z_K(x; z), z)U + \sigma_{n+1,K}(z')W_K$$

and we have that

$$|y_K(U, W, x, z)| \leq |\mu_{n,K}(z_K(x; z))| + |\tilde{\sigma}_{n,K}(z_K(x; z), z)||U| + |\sigma_{n+1,K}(z')||W_K|. \tag{9}$$

Observe that $\tilde{\sigma}_{n,K}$, $\mu_{n,K}(z')$ and $\sigma_{n+1,K}$ are continuous and thus bounded over their compact domains. Thus, the right-hand side of (9) can be written in the form claimed in the statement of this lemma. □

Lemma 3. *The family $\{|h(u, w, x, z)| : x \in \mathbb{X}, z \in \mathbb{Z}\}$ is dominated by an integrable function (with respect to the probability distribution over U and W), i.e. $|h(u, w, x, z)| \leq a(u, w)$, where $\mathbb{E}[a(U, W)] < \infty$. Moreover, $b_r(w) := \max_{\{u:|u|\leq r\}} a(u, w)$ is also integrable for each $r < \infty$.*

Proof of Lemma 3. We will show the statement of the lemma for the case where $c(x) = 1$. The general case follows from this because we can replace $a(u, w)$ by $a(u, w)/\min_{z\in\mathbb{Z}} c(z)$, where the denominator is strictly positive by our assumptions.

Setting $a(U, W) = c_0 + c_1|W_K| + c_2|U|$ using the constants c_0, c_1 and c_2 from Lemma 2 shows that $|h(u, w, x, z)| \leq a(u, w)$, where $\mathbb{E}[a(U, W)] < \infty$.

Moreover, letting $b_r(W) := \max_{\{u:|u|\leq r\}} a(u, w) = c_0 + c_1|W_K| + c_2|r|$, we have that $b_r(W)$ is integrable.

□

Lemma 4. *For each $z \in \mathbb{Z}$, $\hat{\alpha}_{I,J(I)}(z) \to \alpha(z)$ almost surely.*

Proof of Lemma 4. Fix $z \in \mathbb{Z}$. Let

$$\hat{\beta}_I(U; z) = \max_{x\in\mathbb{X}} \frac{1}{J(I)} \sum_{j=1}^{J(I)} h(U, W_j, x, z)$$

and

$$\beta(U; z) = \max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z)|U].$$

Observe that the mapping $(x, u) \to h(u, w, x, z)$ satisfies the assumptions of Lemma A1 in Rubinstein & Shapiro (1993), when u is restricted to the set $\{u : |u| \leq r\}$, i.e.

- $\mathbb{X} \times \{u : |u| \leq r\}$ is compact.
- The mapping $(x, u) \to h(u, w, x, z)$ is continuous for almost every w.
- $h(u, w, x, z) \leq b(w) = \max_{u:|u|\leq r} a(u, w)$ for all $x \in \mathbb{X}$ and $u \in [-r, r]$, and $\mathbb{E}|b(W)| < \infty$ by Lemma 3.

Thus we have by the Lemma A1, w.p. 1, $\frac{1}{J(I)} \sum_{j=1}^{J(I)} h(U, W_j, x, z)$ converges to $\mathbb{E}[h(U, W, x, z)|U]$ uniformly in x, U over $\mathbb{X} \times [-r, r]$.

Choose any $\epsilon > 0$ and let I_1 be large enough on the given sample path that

$$\left| \frac{1}{J(I)} \sum_{j=1}^{J(I)} h(U, W_j, x, z) - \mathbb{E}[h(U, W, x, z)|U] \right| < \epsilon,$$

for all $I \geq I_1$, $U \in [-r, r]$ and $x \in \mathbb{X}$. Then,

$$|\hat{\beta}_I(U; z) - \beta(U; z)| < \epsilon,$$

for all $I \geq I_1$ and $U \in [-r, r]$.
Let

$$U^{(r)} = \begin{cases} U & \text{if} \quad U \in [-r, r] \\ r & \text{if} \quad U > r \\ -r & \text{if} \quad U < -r. \end{cases}$$

and similarly for $U_i^{(r)}$.
Let $\hat{\alpha}_I^{(r)}(z) = \frac{1}{I}\sum_{i=1}^{I} \hat{\beta}_I(U_i^{(r)}; z)$ and $\alpha^{(r)}(z) = \mathbb{E}[\beta(U^{(r)}; z)]$. We have

$$|\hat{\alpha}_I(z) - \alpha(z)| \leq \left|\hat{\alpha}_I(z) - \hat{\alpha}_I^{(r)}(z)\right| + \left|\hat{\alpha}_I^{(r)}(z) - \alpha^{(r)}(z)\right| + \left|\alpha^{(r)}(z) - \alpha(z)\right| \tag{10}$$

For $I \geq I_1$, the second term is bounded above by ϵ. Let's focus on the first and third terms. We have that the third term is bounded above by

$$\begin{aligned} \left|\alpha^{(r)}(z) - \alpha(z)\right| &\leq \mathbb{E}\left[\mathbf{1}_{\{|U|>r\}}|\beta(U) - \beta(U^{(r)})|\right] \\ &\leq \mathbb{E}\left[\mathbf{1}_{\{|U|>r\}}(|\beta(U)| + |\beta(U^{(r)})|)\right] \\ &\leq \mathbb{E}\left[\mathbf{1}_{\{|U|>r\}}(a(U, W) + a(U^{(r)}, W))\right], \end{aligned} \tag{11}$$

where the last inequality comes from

$$\begin{aligned} |\beta(U)| &= |\max_{x \in \mathbb{X}} \mathbb{E}[h(U, W, x, z)|U]| \\ &\leq \max_{x \in \mathbb{X}} \mathbb{E}[a(U, W)|U] = \mathbb{E}[a(U, W)|U]. \end{aligned} \tag{12}$$

Consider the right hand side of (11), we have that

$$\begin{aligned} &\mathbb{E}[\mathbf{1}_{\{|U|>r\}}(a(U, W) + a(U^{(r)}, W))] \\ &\quad = \mathbb{E}\left[\mathbf{1}_{\{|U|>r\}}(2c_0 + 2c_1|W_K| + c_2|U| + |U^{(r)}|)\right] \\ &\quad = 2c_0\mathbb{P}(|U| > r) + 2c_1\mathbb{P}(|U| > r)\mathbb{E}[|W_K|] + c_2\mathbb{E}[\mathbf{1}_{\{|U|>r\}}|U|] + \mathbb{E}[\mathbf{1}_{\{|U|>r\}}|U^{(r)}|] \\ &\quad = 2c_0\mathbb{P}(|U| > r) + 2c_1\mathbb{P}(|U| > r)\mathbb{E}[|W_K|] + c_2\mathbb{E}[\mathbf{1}_{\{|U|>r\}}|U|] + \mathbb{E}[\mathbf{1}_{\{|U|>r\}}|r|] \\ &\quad = 2c_0\mathbb{P}(|U| > r) + 2c_1\mathbb{P}(|U| > r)\mathbb{E}[|W_K|] + c_2\mathbb{E}[\mathbf{1}_{\{|U|>r\}}|U|] + r\mathbb{P}(|U| > r). \end{aligned} \tag{13}$$

Since U is a standard normal random variable, $\mathbb{P}(|U| > r)$ goes to 0 as $r \to \infty$ exponentially fast. Moreover, letting $Y = |U|$, we have that

$$\begin{aligned} \mathbb{E}[\mathbf{1}_{\{|U|>r\}}|U|] &= \mathbb{E}[\mathbf{1}_{\{Y>r\}}Y] \\ &= \int_r^\infty \frac{y}{\sqrt{2\pi}}\left(2\exp\left(-\frac{y^2}{2}\right)\right) dy \\ &= \sqrt{\frac{2}{\pi}}\exp(-\frac{r^2}{2}), \end{aligned}$$

which also converges to 0 as $r \to \infty$. Therefore, the right-hand side of (11) converges to 0 as $r \to \infty$ and thus can be bounded above by ϵ for a large enough r.

Consider the first term. It is bounded above by

$$\begin{aligned} \left|\hat{\alpha}_I(z) - \hat{\alpha}_I^{(r)}(z)\right| &\leq \left|\frac{1}{I}\sum_{i=1}^{I} \mathbf{1}_{\{|U_i|>r\}}\left(\hat{\beta}_I(U_i) - \hat{\beta}_I(U_i^{(r)})\right)\right| \\ &\leq \frac{1}{I}\sum_{i=1}^{I} \mathbf{1}_{\{|U_i|>r\}}\left(|\hat{\beta}_I(U_i)| + |\hat{\beta}_I(U_i^{(r)})|\right) \\ &= \frac{1}{I}\sum_{i=1}^{I} \mathbf{1}_{\{|U_i|>r\}}\frac{1}{J(I)}\sum_{j=1}^{J(I)} a(U_i, W_j) + a(U_i^{(r)}, W_j) \end{aligned} \tag{14}$$

By the strong law of large number the term on the right hand side of (14) converges almost surely to the term on the right hand side of (11). Since we have chosen r large enough such that the right hand side of (11) is bounded by ϵ, we can choose I large enough such that the right hand side of (14) is also bounded by ϵ.

Therefore, for a given $\epsilon > 0$, w.p. 1,

$$|\hat{\alpha}_{I,J(I)}(z) - \alpha(z)| < 3\epsilon,$$

for large enough I. This implies $\lim_{I\to\infty} \hat{\alpha}_{I,J(I)}(z) = \alpha(z)$ almost surely as desired. □

Lemma 5. *$\alpha(z)$ is continuous in z and $\hat{\alpha}_{I,J(I)}(z)$ converges uniformly to $\alpha(z)$ on $\mathbb{Z}$ as $I \to \infty$ and $\lim_{I\to\infty} J(I) = \infty$.*

Proof of Lemma 5. Observe that

$$\max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z)|U] \leq \max_{x\in\mathbb{X}} \mathbb{E}[a(U, W)|U] = \mathbb{E}[a(U, W)|U], \tag{15}$$

where the inequality follows from Lemma 3. The term of the right hand side of (15) is an integrable function of U since

$$\mathbb{E}[|\mathbb{E}[a(U, W)|U|] = \mathbb{E}[\mathbb{E}[a(U, W)|U]] = \mathbb{E}[a(U, W)] < \infty. \tag{16}$$

Consider any $z \in \mathbb{Z}$ and a sequence $\{z_k : k \geq 1\} \subseteq \mathbb{Z}$ converging to z. By the Lebesgue dominated convergence theorem and (16),

$$\lim_{k\to\infty} \mathbb{E}[\max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z_k)|U]] = \mathbb{E}[\lim_{k\to\infty} \max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z_k)|U]] \tag{17}$$

Fixing U, let $g(x, z) = \mathbb{E}[h(U, W, x, z)|U]$. This function is continuous jointly in x, z because for an arbitrary sequence $\{(x_k, z_k) : k \geq 1\}$ converging to (x, z), we have

$$\lim_{k\to\infty} \mathbb{E}[h(U, W, x_k, z_k)] = \mathbb{E}[\lim_{k\to\infty} h(U, W, x_k, z_k)] = \mathbb{E}[h(U, W, x, z)],$$

where we have used Lemma 3 and the Lebesgue dominated convergence theorem in the first equality and used continuity (Lemma 1) in the second equality.

Since $\mathbb{X} \times \mathbb{Z}$ is compact, g is uniformly continuous over $\mathbb{X} \times \mathbb{Z}$. Thus, there exists a large enough K such that

$$|g(x, z_k) - g(x, z)| < \epsilon,$$

for all $k \geq K$ and all $x \in \mathbb{X}$. Let $x_k^* \in \arg\max_{x\in\mathbb{X}} g(x, z_k)$. For all such k, we have

$$\max_{x\in\mathbb{X}} g(x, z_k) - \max_{x\in\mathbb{X}} g(x, z) \leq g(x_k^*, z_k) - g(x_k^*, z) < \epsilon. \tag{18}$$

Similarly, let $x^* \in \arg\max_{x\in\mathbb{X}} g(x, z)$. For all such k, we have

$$-\epsilon < g(x^*, z_k) - g(x^*, z) \leq \max_{x\in\mathbb{X}} g(x, z_k) - \max_{x\in\mathbb{X}} g(x, z) \tag{19}$$

Equations (18) and (19) imply

$$\left|\max_{x\in\mathbb{X}} g(x, z_k) - \max_{x\in\mathbb{X}} g(x, z)\right| < \epsilon.$$

This shows that $\lim_{k\to\infty} \max_{x\in\mathbb{X}} g(x, z_k) = \max_{x\in\mathbb{X}} g(x, z)$. Thus, (17) becomes

$$\begin{aligned}\lim_{k\to\infty} \mathbb{E}[\max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z_k)|U]] &= \mathbb{E}[\lim_{k\to\infty} \max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z_k)|U]] \\ &= \mathbb{E}[\max_{x\in\mathbb{X}} \mathbb{E}[h(U, W, x, z)|U]] = \alpha(z).\end{aligned} \tag{20}$$

Equation (20) implies $\lim_{k\to\infty} \alpha(z_k) = \alpha(z)$, showing that $\alpha(\cdot)$ is continuous on $\mathbb{Z}$.

Now consider a sequence $\mathbb{N}_k$ of neighborhoods of a generic point $z \in \mathbb{Z}$ shrinking to $\{z\}$. Let

$$b_k(u, w) = \sup\{|h(u, w, x, z) - h(u, w, x, y)| : y \in \mathbb{N}_k, x \in \mathbb{X}\}.$$

Recall that $h(u, w, \cdot, \cdot)$ is continuous for almost every (u, w). Choose u, w such that $h(u, w, \cdot, \cdot)$ is continuous on $\mathbb{X} \times \mathbb{Z}$. Since $\mathbb{X} \times \mathbb{Z}$ is compact, $h(u, w, \cdot, \cdot)$ is uniformly continuous on $\mathbb{X} \times \mathbb{Z}$. Therefore, given $\epsilon > 0$, there exists $\delta > 0$ such that $|h(u, w, x', z') - h(u, w, x'', z'')| < \epsilon$ when $||(x', z') - (x'', z'')|| < \delta$. Thus, for a large enough k, we have $|h(u, w, x, z) - h(u, w, x, y)| < \epsilon$ for all $y \in \mathbb{N}_k$. This implies $\lim_{k\to\infty} b_k(u, w) = 0$ for almost every (u, w).

Define

$$c_k(U_i, W_{1:J}) = \sup \left\{ \left| \max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z) - \max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z') \right| : z' \in \mathbb{N}_k \right\}.$$

Let $x^*(z') \in \arg\max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z')$. For $z' \in \mathbb{N}_k$,

$$\begin{aligned}
&\max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z) - \max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z') \\
&\qquad \le \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x^*(z), z) - \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x^*(z), z') \\
&\qquad = \frac{1}{J} \sum_{j=1}^{J} [h(U_i, W_j, x^*(z), z) - h(U_i, W_j, x^*(z), z')] \\
&\qquad \le \frac{1}{J} \sum_{j=1}^{J} |h(U_i, W_j, x^*(z), z) - h(U_i, W_j, x^*(z), z')| \\
&\qquad \le \frac{1}{J} \sum_{j=1}^{J} b_k(U_i, W_j). \qquad (21)
\end{aligned}$$

Similarly,

$$\begin{aligned}
&\max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z) - \max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z') \\
&\qquad \ge \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x^*(z'), z) - \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x^*(z'), z') \\
&\qquad \ge -\frac{1}{J} \sum_{j=1}^{J} |h(U_i, W_j, x^*(z'), z) - h(U_i, W_j, x^*(z'), z')| \\
&\qquad \ge -\frac{1}{J} \sum_{j=1}^{J} b_k(U_i, W_j). \qquad (22)
\end{aligned}$$

Equations (21) and (22) imply

$$\left| \max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z) - \max_{x\in\mathbb{X}} \frac{1}{J} \sum_{j=1}^{J} h(U_i, W_j, x, z') \right| \le \frac{1}{J} \sum_{j=1}^{J} b_k(U_i, W_j).$$

This implies

$$c_k(U_i, W_{1:J}) \le \frac{1}{J} \sum_{j=1}^{J} b_k(U_i, W_j).$$

Thus for all $z' \in \mathbb{N}_k$,

$$|\hat{\alpha}_{I,J(I)}(z') - \hat{\alpha}_{I,J(I)}(z)| \leq \frac{1}{I}\sum_{i=1}^{I} c_k(U_i, W_{1:J}) \leq \frac{1}{I}\sum_{i=1}^{I}\frac{1}{J}\sum_{j=1}^{J} b_k(U_i, W_j). \tag{23}$$

Fix J to be a function of I, i.e. $J = J(I)$ where $\lim_{I\to\infty} J(I) = \infty$. By the strong law of large number, the right hand side of Equation (23) converges to $\mathbb{E}[b_k(U, W)]$ as $I \to \infty$ almost surely.

Lemma 3 implies $b_k(U, W) \leq 2a(U, W)$. Thus, by the Lebesgue dominated convergence theorem, $\lim_{k\to\infty} \mathbb{E}[b_k(U, W)] = \mathbb{E}[\lim_{k\to\infty} b_k(U, W)] = 0$.

For any $\epsilon > 0$, there is a large enough K such that $\mathbb{E}[b_K(U, W)] < \epsilon$. Moreover, w.p.1, there is a large enough I^* such that

$$\sup_{z'\in\mathbb{N}_K} |\hat{\alpha}_{I,J(I)}(z') - \hat{\alpha}_{I,J(I)}(z)| < \epsilon \quad \forall I \geq I^*.$$

Moreover, since α is continuous, it is possible to choose K large enough that

$$\sup_{z'\in\mathbb{N}_K} |\alpha(z') - \alpha(z)| < \epsilon.$$

We have shown that, for any point $z \in \mathbb{Z}$ and any $\epsilon > 0$, there is a neighborhood $\mathbb{N}(z, \epsilon)$ of z such that w.p. 1 there is a large enough I^* such that

$$\sup_{z'\in\mathbb{N}(z,\epsilon)} |\hat{\alpha}_{I,J(I)}(z') - \hat{\alpha}_{I,J(I)}(z)| < \epsilon \quad \forall I \geq I^*$$

and that

$$\sup_{z'\in\mathbb{N}(z,\epsilon)} |\alpha(z') - \alpha(z)| < \epsilon.$$

Let $\{\mathbb{N}(z, \epsilon) : z \in \mathbb{Z}\}$ be an open cover of $\mathbb{Z}$. Since $\mathbb{Z}$ is compact, we can choose a finite subcover. This gives a collection of points $z_1, \ldots, z_J$ with neighborhoods $\mathbb{N}(z_j, \epsilon)$ that cover $\mathbb{Z}$ such that w.p. 1, for sufficiently large I,

$$\sup\{|\hat{\alpha}_{I,J(I)}(z') - \hat{\alpha}_{I,J(I)}(z_j)| : z' \in \mathbb{N}(z_j, \epsilon)\} < \epsilon$$

and

$$\sup\{|\alpha(z') - \alpha(z_j)| : z' \in \mathbb{N}(z_j, \epsilon)\} < \epsilon.$$

By Lemma 4, we have

$$|\hat{\alpha}_{I,J(I)}(z_j) - \alpha(z_j)| < \epsilon.$$

Thus, given $\epsilon > 0$, w.p. for I large enough,

$$|\hat{\alpha}_{I,J(I)}(z) - \alpha(z)| \leq |\hat{\alpha}_{I,J(I)}(z) - \hat{\alpha}_{I,J(I)}(z_j)| + |\hat{\alpha}_{I,J(I)}(z_j) - \alpha(z_j)| + |\alpha(z_j) - \alpha(z)| < 3\epsilon,$$

where $z \in \mathbb{N}(z_j, \epsilon)$. This implies that $\hat{\alpha}_{I,J(I)}(z) \to \alpha(z)$ uniformly on $\mathbb{Z}$ as $I \to \infty$, as desired. □

Using the above lemmas, we are now in position to prove Theorem 1.

Proof of Theorem 1. From Lemma 5, we know that for any $\epsilon > 0$ and a sufficiently large I and all $z \in \mathbb{Z}$,

$$|\hat{\alpha}_{I,J(I)}(z) - \alpha(z)| < \epsilon \text{ w.p. 1.} \tag{24}$$

Let $\hat{z}_{I,J(I)} \in \arg\max_{z\in\mathbb{Z}} \hat{\alpha}_{I,J(I)}(z)$ and $z^* \in \arg\max_{z\in\mathbb{Z}} \alpha(z)$. It follows that

$$|\hat{\alpha}_{I,J(I)}(\hat{z}_{I,J(I)}) - \alpha(\hat{z}_{I,J(I)})| < \epsilon,$$

which implies

$$\hat{\alpha}_{I,J(I)}(\hat{z}_{I,J(I)}) < \alpha(\hat{z}_{I,J(I)}) + \epsilon < \alpha(z^*) + \epsilon. \tag{25}$$

Similarly, we have

$$|\hat{\alpha}_{I,J(I)}(z^*) - \alpha(z^*)| < \epsilon,$$

which implies

$$-\epsilon + \alpha(z^*) < \hat{\alpha}_{I,J(I)}(z^*) < \hat{\alpha}_{I,J(I)}(\hat{z}_{I,J(I)}) \tag{26}$$

Equations (25) and (26) yield $|\hat{\varphi}_{I,J(I)} - \varphi^*| < \epsilon$. Since ϵ is arbitrary, this implies $\hat{\varphi}_{I,J(I)}$ converges to φ^* almost surely as desired.

Furthermore, suppose that z^* is a unique maximizer of $\alpha(z)$ over $\mathbb{Z}$. Consider any neighborhood $\mathbb{N}$ of $z^* \in \mathbb{Z}$. Following from the continuity of α and the compactness of $\mathbb{Z}$, there exists $\epsilon > 0$ such that

$$\alpha(z) > \alpha(z^*) + 2\epsilon,$$

for all $z \in \mathbb{Z}$ and $z \notin \mathbb{N}$. This combines with (24) implies

$$\hat{\alpha}_{I,J(I)}(z) > \alpha(z^*) + \epsilon, \tag{27}$$

for all $z \in \mathbb{Z}$ and $z \notin \mathbb{N}$. Also, since we have $|\hat{\varphi}_{I,J(I)} - \varphi^*| < \epsilon$, this implies

$$\hat{\alpha}_{I,J(I)}(\hat{z}_{I,J(I)}) = \hat{\varphi}_{I,J(I)} < \varphi^* + \epsilon = \alpha(z^*) + \epsilon. \tag{28}$$

Equation (28) implies $\hat{z}_{I,J(I)} \in \mathbb{N}$. Since $\mathbb{N}$ can be chosen arbitrarily, it implies $\hat{z}_{I,J(I)} \to z^*$ almost surely.

Now, let us consider the case where α has multiple maximizers over $\mathbb{Z}$. Suppose for the sake of contradiction that $d(\hat{z}_{I,J(I)}, \mathcal{Z}^*) := \inf_{z \in \mathcal{Z}^*} ||\hat{z}_{I,J(I)} - z|| \nrightarrow 0$. Since $\mathbb{Z}$ is compact, there exists a sequence $\{\hat{z}_{I,J(I)} : I \geq 1\}$ such that $d(\hat{z}_{I,J(I)}, \mathcal{Z}^*) \geq \epsilon$ for some $\epsilon > 0$ and the sequence $\{\hat{z}_{I,J(I)} : I \geq 1\}$ converges to a point $z' \in \mathbb{Z}$, but not in $\mathcal{Z}^*$. By the continuity of α, we have that $\alpha(\hat{z}_{I,J(I)}) \to \alpha(z') < \varphi^*$ as $I \to \infty$. Moreover, $\hat{\varphi}_{I,J(I)} = \hat{\alpha}_{I,J(I)}(\hat{z}_{I,J(I)})$ and

$$\hat{\alpha}_{I,J(I)}(\hat{z}_{I,J(I)}) - \alpha(z') = (\hat{\alpha}_{I,J(I)}(\hat{z}_{I,J(I)}) - \alpha(\hat{z}_{I,J(I)})) + (\alpha(\hat{z}_{I,J(I)}) - \alpha(z')). \tag{29}$$

The first term on the right-hand side of (29) goes to zero as $I \to \infty$ by Lemma 5, and the second term also converges to 0 by the continuity of α. This implies that $\hat{\varphi}_{I,J(I)} \to \alpha(z') < \varphi^*$, which is a contradiction. Thus, we have $d(\hat{z}_{I,J(I)}, \mathcal{Z}^*) \to 0$ almost surely, as desired. □

C. Additional Details on Acquisition Function Optimization

We describe model configuration and hyperparameters used to optimize the acquisition functions used in our numerical experiments. We also present a comparison between the one-shot optimization and optimization-via-discretization approaches for optimizing the p-KGFN acquisition.

C.1. Hyperparameters in Acquisition Function Optimization

In our experiments, all methods utilize independent GPs with zero mean functions and the Mateŕn 5/2 kernel (Genton, 2001), with automatic relevance determination (ARD). The lengthscales of the GPs are assumed to have Gamma priors: for the Ackley, FreeSolv and Pharm problems, Gamma(3, 6); for the Manu-GP problem, Gamma(5, 2). The outputscale parameters are assumed to have Gamma(2, 0.15) priors in all problems. The lengthscales and outputscales are then estimated via maximum a posteriori (MAP) estimation.

In p-KGFN, we estimate the posterior mean of a function network's final node value with $J = 64$ quasi-MC samples using Sobol sequences (Balandat et al., 2020). For EIFN, we follow the implementation in Astudillo & Frazier (2021a), using $J = 128$.

To compute MC estimates for the p-KGFN acquisition value, we use $I = 8$ fantasy models. As described in Section 5.4, we optimize the p-KGFN acquisition function via discretization, replacing the domain of the optimization problem in line 6 of Algorithm 2 by a discrete set of candidate solutions $\mathbb{A}$. We include in $\mathbb{A}$ the current maximizer of the final node posterior

mean x_n^*, $N_T = 10$ points obtained from a Thompson sampling method described in the main text, $N_L = 10$ local points with $r = 0.1$.

When optimizing EIFN, TSFN, KGFN with full evaluations or p-KGFN for a problem with d-dimensional function network input, we use $100d$ raw samples for identifying the starting points for the multi-start acquisition optimization and $10d$ starting points for the multi-start acquisition function optimization. We set the number of raw samples to 100 and the number of starting points to 20 when optimizing the standard EI and KG acquisition functions.

We refit the hyperparameters of the GP models in each iteration. For p-KGFN, this occurs after we obtain one additional observation of a specific function node; for other benchmarks, this occurs after we obtain observations of the entire function network for a design point.

All algorithms are implemented in the open source BoTorch package (Balandat et al., 2020). We extend the implementation outlined in Astudillo & Frazier (2021a) to enable partial evaluations for function networks.

C.2. Comparison Between One-Shot Optimization and Optimization-Via-Discretization

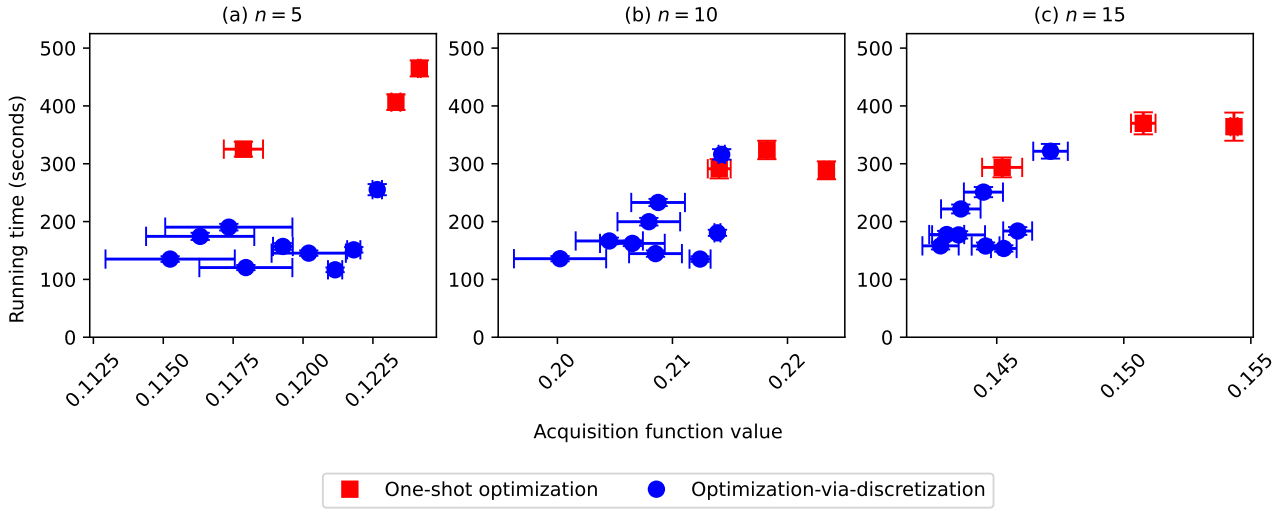

Figure 6: Comparison between one-shot optimization and optimization-via-discretization in terms of acquisition value attained and computation time, on the Ackley6D problem with different numbers of fully-evaluated points: (a) $n = 5$, (b) $n = 10$, and (c) $n = 15$. Both the value averaging over 100 trials and the standard error are presented.

We perform a comparative analysis between two methods for optimizing KG-based acquisition functions discussed in Section 5.4: "One-shot" optimization and optimization-via-discretization.

The one-shot optimization method, proposed by Balandat et al. (2020), has gained popularity in optimizing non-myopic acquisition functions in recent years (Jiang et al., 2020; Astudillo et al., 2021; Daulton et al., 2023). This method effectively addresses the nested optimization problem, a common challenge encountered when optimizing KG. However, it introduces its own challenges – namely, by turning the nested optimization problem into a single high-dimensional (and typically more difficult) optimization problem, the one-shot optimization approach can be more likely to get stuck in local optima and require long computation time (due to the higher number of optimization variables, and the potential of the numerical optimization failing to converge).

Optimization-via-discretization, as used by Frazier et al. (2009), is a commonly used alternative technique for optimizing KG (Ungredda et al., 2022; Buckingham et al., 2023). This method accelerates KG optimization by discretizing the domain of the inner optimization problem. Nevertheless, it has its own limitations, as the discretization will lead to less accurate solutions for the inner optimization problem, potentially resulting in sub-optimal outcomes, especially in higher dimensions (due to the coarser coverage of the space).

For p-KGFN, we follows the optimization-via-discretization approach. However, rather than selecting points randomly within the domain or arranging them in a grid, we intelligently select the points that form the discrete set for the inner

optimization problem (see Section 5.4).

To better understand the performance of one-shot optimization and optimization-via-discretization in the context of optimizing p-KGFN, we conduct a comparative analysis. Specifically, we examine both approaches under different configurations using the Ackley6D test problem. We consider problem instances where we have access 5, 10, and 15 full evaluations of the function network. These observations are held fixed and used to fit GP network models. We seek to optimize p-KGFN at the first function node.

For both approaches, we explore three choices ($I = 2, 4, 8$) for the number of fantasy models for estimating the p-KGFN acquisition values. In the optimization-via-discretization approach, we generate the discrete set by using Thompson sampling approach, i.e., sampling N_T realizations of a function network based on the current posterior distributions and optimize the realizations to obtain their maximizers to include in the discrete set $\mathbb{A}$, and randomly choosing N_L local points close to the current posterior mean maximizer x_n^*. The current posterior mean maximizer itself is also included in the set. We consider three discrete set sizes for the inner optimization problem: $|\mathbb{A}| = 11$ ($N_T = N_L = 5$), $|\mathbb{A}| = 21$ ($N_T = N_L = 10$), and $|\mathbb{A}| = 41$ ($N_T = N_L = 20$). We measure performance in terms of both true acquisition function value of the selected candidates (approximated by the MC estimator described in Algorithm 2, with $I = 512$ and $J = 128$) and computational time (measured in seconds), averaging over 100 trials.

In Figure 6 we observe that although one-shot optimization demonstrates the best performance in terms of acquisition value attained, optimization-via-discretization can attain comparable acquisition value at significantly reduced run times. For the setup used in our experiments ($I = 8$ and $|\mathbb{A}| = 21$), optimization-via-discretization achieves slightly lower acquisition value compared to the best-performing one-shot optimization method with $I = 8$ (1.89%, 4.25%, and 5.51% lower for the problem instances with 5, 10, and 15 fully evaluated points, respectively). However, it significantly reduces the average run time compared to the best-performing one-shot optimization algorithm, by 67.5%, 37.4%, and 50.5%. The significant reduction in run time, along with the marginal loss in acquisition values, justifies the employment of the discretization approach in our experiments. The full comparison analysis results are summarized in Table 1.

D. Additional Details on Numerical Experiments

D.1. Ackley6D (Ackley)

We design the Ackley synthetic test problem as a two-stage function network (Figure 3a). The first function node is the 6-dimensional negated Ackley function (Ackley, 2012):

$$f_1(x) = 20 \exp\left(-0.2\sqrt{\frac{1}{6}\sum_{i=1}^{6} x_i^2}\right) + \exp\left(\frac{1}{6}\sum_{i=1}^{6}\cos(2\pi x_i)\right) - 20 - \exp(1),$$

where $x_i \in [-2, 2]$ for $i = 1, \ldots, 6$. The second function node, which takes as an input the output of the first function node, is defined as follows

$$f_2(y) = -y \sin\left(\frac{5y}{6\pi}\right).$$

D.2. Manufacturing Network (Manu-GP)

Motivated by real-world manufacturing applications where several intermediate parts are produced and then assembled to create a final product, we formulate this second test problem involving two processes, each of which takes a two-dimensional input (Figure 3b). The first process consists of two sequential sub-processes, denoted as f_1 and f_2. The first sub-process takes x_1, x_2 as inputs and returns y_1 which is taken by the second sub-process to produce an output y_2. The second process, f_3, takes x_3 and x_4 as input and produces an output y_3. The outputs y_2 and y_3 are then combined in the final process f_4 to produce a final output y_4, which we aim to maximize. This experiment is designed to mimic a manufacturing scenario. Here, we set each input $x_i \in [-1, 1]$, for $i = 1, \ldots, 4$.

For each function node, we draw a sample path from a GP prior with the Matérn 5/2 kernel (Genton, 2001). Notably, we choose different lengthscales in the kernels for difference function nodes: 0.631 for f_1, 1 for f_2, 1 for f_3, and 3 for f_4. The outputscale parameter in the kernel is set to 0.631 for all functions but f_4, which has outputscale equal to 10. We show the drawn sample paths for each of the function nodes in Figure 7.

Table 1: Comparisons between one-shot optimization and optimization-via-discretization on the Ackley6D test problem, with 5, 10, and 15 design points fully evaluated across the network. Performances are reported in terms of acquisition function value obtained and running time. Both the value averaging over 100 trials and the standard error are reported. Numbers in boldface denote the highest acquisition function value and the shortest running time.

(a) Number of fully evaluated points: 5

Method	**Num of fantasy models I**	**Size of discrete set $\|\mathbb{A}\|$**	**Avg. acqf. val.**	**Avg. running time (seconds)**
One-shot	2	NA	0.11787 ± 0.00070	325 ± 13
	4	NA	0.12332 ± 0.00016	406 ± 13
	8	NA	$\mathbf{0.12416 \pm 0.00004}$	465 ± 14
Discretization	2	11	0.11525 ± 0.00231	135 ± 5
	2	21	0.11928 ± 0.00035	157 ± 6
	2	41	0.1164 ± 0.00194	174 ± 6
	4	11	0.11795 ± 0.00166	120 ± 5
	4	21	0.12020 ± 0.00133	145 ± 5
	4	41	0.11734 ± 0.00226	190 ± 6
	8	11	0.12115 ± 0.00026	$\mathbf{117 \pm 4}$
	8	21	0.12181 ± 0.00024	151 ± 5
	8	41	0.12265 ± 0.00017	255 ± 10

(b) Number of fully evaluated points: 10

Method	**Num of fantasy models I**	**Size of discrete set $\|\mathbb{A}\|$**	**Avg. acqf. val.**	**Avg. running time (seconds)**
One-shot	2	NA	0.2141 ± 0.0010	291 ± 17
	4	NA	0.2182 ± 0.0006	324 ± 16
	8	NA	$\mathbf{0.2234 \pm 0.0002}$	289 ± 16
Discretization	2	11	0.2085 ± 0.0023	145 ± 5
	2	21	0.2045 ± 0.0029	166 ± 6
	2	41	0.2079 ± 0.0027	200 ± 7
	4	11	0.2002 ± 0.0040	136 ± 5
	4	21	0.2065 ± 0.0028	162 ± 5
	4	41	0.2087 ± 0.0023	233 ± 6
	8	11	0.2124 ± 0.0009	$\mathbf{135 \pm 5}$
	8	21	0.2139 ± 0.0003	181 ± 5
	8	41	0.2143 ± 0.0003	316 ± 9

(c) Number of fully evaluated points: 15

Method	**Num of fantasy models I**	**Size of discrete set $\|\mathbb{A}\|$**	**Avg. acqf. val.**	**Avg. running time (seconds)**
One-shot	2	NA	0.1452 ± 0.0008	294 ± 17
	4	NA	0.1508 ± 0.0005	370 ± 19
	8	NA	$\mathbf{0.1543 \pm 0.0001}$	364 ± 24
Discretization	2	11	0.1428 ± 0.0007	158 ± 6
	2	21	0.1430 ± 0.0006	177 ± 6
	2	41	0.1436 ± 0.0008	222 ± 8
	4	11	0.1446 ± 0.0005	157 ± 6
	4	21	0.1435 ± 0.0010	177 ± 6
	4	41	0.1445 ± 0.0008	250 ± 9
	8	11	0.1453 ± 0.0005	$\mathbf{153 \pm 6}$
	8	21	0.1458 ± 0.0006	184 ± 7
	8	41	0.1471 ± 0.0007	321 ± 13

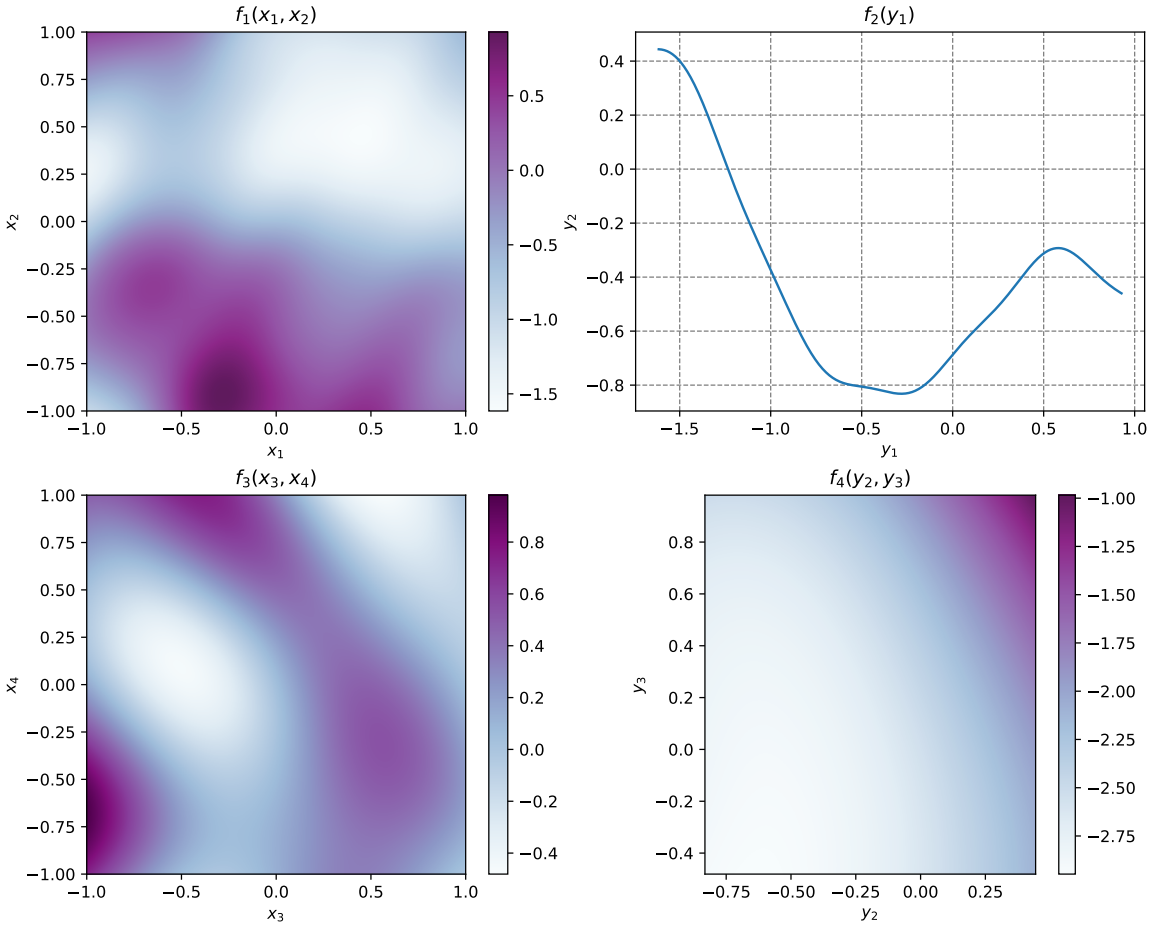

Figure 7: The sample paths drawn from GP priors for the manufacturing problem.

D.3. Molecular Design (FreeSolv)

We use the FreeSolv dataset from Mobley & Guthrie (2014), which comprises both calculated and experimental free energies (in kcal/mol) for 642 small molecules. Since lower free energy is preferred, we negate free energy (both calculated and experimental), setting our objective to maximizing the negative experimental free energy.

To represent each small molecule in a continuous space, we utilize a variational autoencoder trained on the Zinc dataset as studied in Gómez-Bombarelli et al. (2018), resulting in a 196-dimensional representation in the unit cube, i.e., $x \in [0, 1]^{196}$. We reduce the dimentionality of the representation to three through the standard principal component analysis (PCA) technique.

We then utilize the entire dataset to train two GP models, and use the posterior mean of the trained model as the underlying functions in the function network (Figure 3a):

1. f_1 takes a three-dimensional representation as input and predicts the negative calculated free energy y_1;

2. f_2 takes the negative calculated free energy as input and predicts the negative experimental free energy (Figure 8).

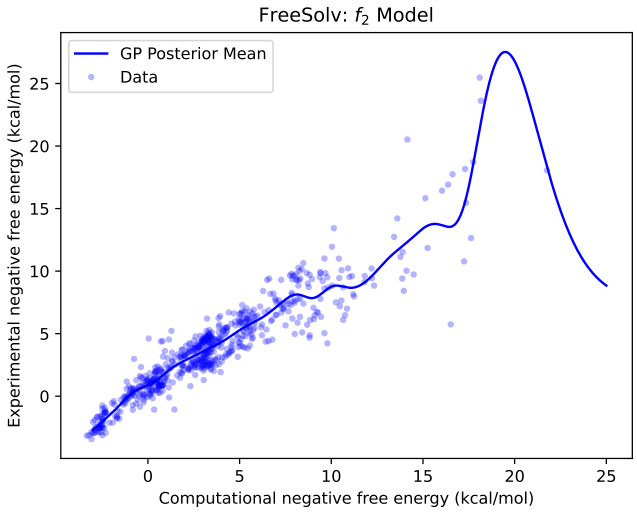

Figure 8: The posterior mean function of the GP fitted between calculated and experimental free energies.

D.4. Pharmaceutical Product Development (Pharm)

In this test problem, we tackle the challenge of manufacturing orally disintegration tablets (ODTs) that meet pharmaceutical standards, focusing on two crucial properties: disintegration time (f_1) and tensile strength (f_2). To model these properties, we adopt the neural network (NN) models proposed in Sano et al. (2020).

Specifically, the two properties are defined as functions of four parameters describing the production process, namely, β form D-mannitol ratio in the total D-mannitol (x_1), L-HPC ratio in a formulation (x_2), granulation fluid level (x_3) and compression force (x_4). Each parameter x_i is constrained to the range $[-1, 1]$. The fitted NN models are shown as follows:

$$
\begin{aligned}
f_1(x) = &-3.95 + 9.20 \times (1 + \exp(-(0.32 + 5.06x_1 - 4.07x_2 - 0.36x_3 - 0.34 \times x_4))^{-1} \\
&+ 9.88 \times (1 + \exp(-(-4.83 + 7.43x_1 + 3.46x_2 + 9.19x_3 + 16.58x_4)))^{-1} \\
&+ 10.84 \times (1 + \exp(-7.90 - 7.91x_1 - 4.48x_2 - 4.08x_3 - 8.28x_4))^{-1} \\
&+ 15.18 \times (1 + \exp(-(9.41 - 7.99x_1 + 0.65x_2 + 3.14x_3 + 0.31x_4)))^{-1}.
\end{aligned}
$$

and

$$\begin{aligned} f_2(x) = 1.07 &+ 0.62 \times (1 + \exp(-(3.05 + 0.03x_1 - 0.16x_2 + 4.03x_3 - 0.54x_4)))^{-1} \\ &+ 0.65 \times (1 + \exp(-(1.78 + 0.60x_1 - 3.19x_2 + 0.10x_3 + 0.54x_4)))^{-1} \\ &- 0.72 \times (1 + \exp(-(0.01 + 2.04x_1 - 3.73x_2 + 0.10x_3 - 1.05x_4)))^{-1} \\ &- 0.45 \times (1 + \exp(-(1.82 + 4.78x_1 + 0.48x_2 - 4.68x_3 - 1.65x_4)))^{-1} \\ &- 0.32 \times (1 + \exp(-(2.69 + 5.99x_1 + 3.87x_2 + 3.10x_3 - 2.17x_4)))^{-1}, \end{aligned}$$

To measure the quality of a tablet, the same study introduced a score function f_3 (treated as a known function in our experiment), which combines the two properties f_1 and f_2:

$$f_3 = \frac{(60 - f_1)}{60} \times \frac{f_2}{1.5},$$

where the first term aims to ensure that the disintegration time is not too long (less than 60 seconds), and the second term aims to ensure that the tensile strength is large enough for production and distribution.

D.5. Additional Experiment Without Upstream Evaluation Condition: Ackley6D+Matyas2D (AckMat)

We consider the setting in which node evaluations do not require previously evaluated inputs from upstream nodes. Instead, we assume each node can be evaluated at any point in the set of possible outputs of the upstream nodes. We design this problem as a 7-dimensional cascade network where the first node is the 6-dimensional Ackley function (Ackley, 2012):

$$f_1(x) = -20\exp\left(-0.2\sqrt{\frac{1}{6}\sum_{i=1}^{6} x_i^2}\right) - \exp\left(\frac{1}{6}\sum_{i=1}^{6}\cos\left(2\pi x_i\right)\right) + 20 + \exp(1),$$

where $x_i \in [-2, 2]$ for $i = 1, \ldots, 6$. The second function node, which takes as input the output y of the first node and one additional input x_7, is the negated Matyas function (Jamil & Yang, 2013):

$$f_2(y, x_7) = -0.26(y^2 + x_7^2) + 0.48yx_7.$$

We set the range $x_7 \in [-10, 10]$ and we assume that the range of the output from the first node is known, i.e. $y \in [0, 20]$. The evaluation costs for this experiment are set to be $c_1 = 1$ and $c_2 = 49$ and we restrict to the same BO budget equal to 700. The results in Figure 9 show that p-KGFN is also effective in this setting. Interestingly, we see that p-KGFN makes progress more slowly in the beginning, but then quickly overtakes and substantially outperform all baselines. This reflects the fact that the algorithm initially allocates most of its budget to learning about the behavior of the first node that is cheap to evaluate, and then with that knowledge moves to effectively optimize the second.

D.6. Additional Experiment with Noisy Observations

In this section, we consider the FreeSolv problem presented in Section 6 and Appendix D.3. We conduct additional experiments that add normally distributed noise to a node's output before it is passed to subsequent nodes. We assume that the noise at each function node follows the standard normal distribution $\mathcal{N}(0, 1)$.

We use the noisy observations to update the GP describing each node. This entails standard equations for Gaussian process regression with noisy observations (Williams & Rasmussen, 2006).

We consider our default setting, i.e. costs $c_1 = 1$ and $c_2 = 49$ with a total BO budget equal to 700. Figure 10 illustrates the performance comparison between p-KGFN and benchmark algorithms on this variant of the test problem. The results demonstrate that p-KGFN still outperforms all benchmark algorithms, indicating its robustness to observation noise.

D.7. Additional Experiments where Downstream Nodes are More Difficult to Optimize

We conduct additional experiments where upstream nodes are more difficult to optimize than downstream nodes. We consider two problem setups:

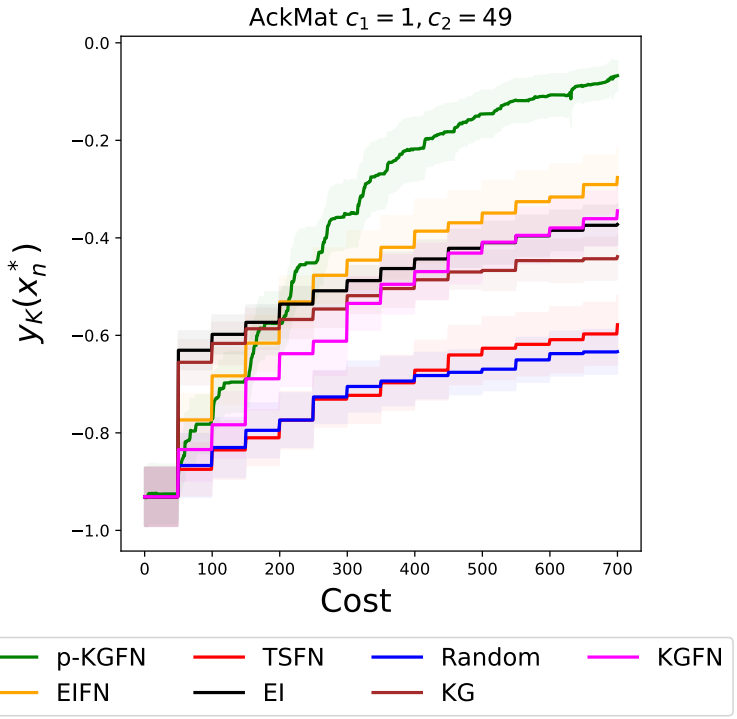

Figure 9: Optimization performance on AckMat problem without upstream evaluation condition comparing between our proposed p-KGFN and benchmarks including EIFN, KGFN, TSFN, EI, KG and Random.

- (GPs-1): A sequential network with two nodes, shown in Figure 3a. The first function node takes in a 1-D input, $x \in [-1, 1]$. Both function nodes are drawn from GP priors. The lengthscales of the GP priors for the first and second nodes are set to 0.5 and 0.25, respectively. This ensures that the second node is more difficult to optimize compared to the first node. As with the test problems in our main paper, we set the cost of evaluating the second node to be substantially higher than that of evaluating the first node, i.e., $c_1 = 1$ and $c_2 = 49$. We set the total BO budget at 700.

- (GPs-2): A function network with four function nodes (Figure 11). The first three function nodes, f_1, f_2 and f_3, respectively take input $x_1, x_2, x_3 \in [-1, 1]$. The final function node, f_4, takes the outputs of f_1, f_2, and f_3 as its inputs. All functions are drawn from GP priors with a common lengthscale. We set the evaluation costs to be $c_1 = c_2 = c_3 = 1$ and $c_4 = 47$ and a total BO budget of 700.

We used the same settings for other parameters, such as the number of initial observations, as in the main experiments.

As presented in Figure 12, our algorithm, p-KGFN, performs comparably to the other benchmarks in these additional problems.

E. Sensitivity Analysis for Evaluation Costs

We conduct a sensitivity analysis to examine the impact of cost functions on the optimization performance across three experiments with two nodes presented in the main text: Ackley (result is presented in the main text), FreeSolv and Pharm. In this section, we again consider scenarios where evaluating a downstream node requires previously obtained outputs from its parent nodes. This implies that the first node must be evaluated regardless of its cost. Our focus is thereby directed towards assessing the consequences of varying the cost associated with the second node. We investigate three cost function scenarios: (a) $c_1 = 1, c_2 = 1$; (b) $c_1 = 1, c_2 = 10$; and (c) $c_1 = 1, c_2 = 49$, which correspond to the situations where both nodes have similar evaluation costs, where one node has a higher evaluation cost than the other, and where one node has an exceptionally high evaluation cost, respectively. The evaluation budgets for each problem are set to 50, 150, and 700, respectively, in the three scenarios.

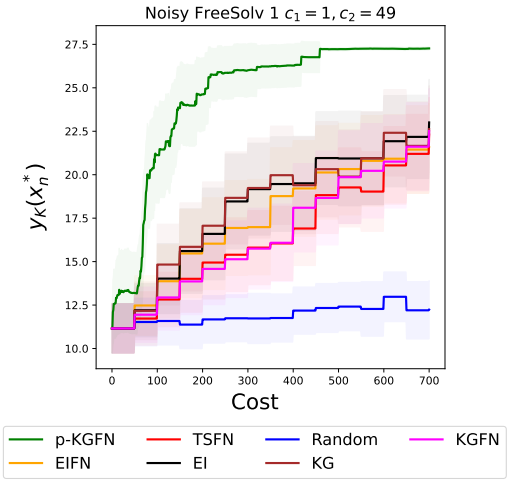

Figure 10: Optimization performance comparing between our proposed p-KGFN and benchmarks including EIFN, KGFN, TSFN, EI, KG, and Random on the FreeSolv problem with noisy observations, averaging over 30 trials.

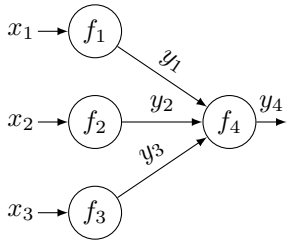

Figure 11: A function network structure for an additional experiment where the first layer node is harder-to-learn than the second layer (GPs-2).

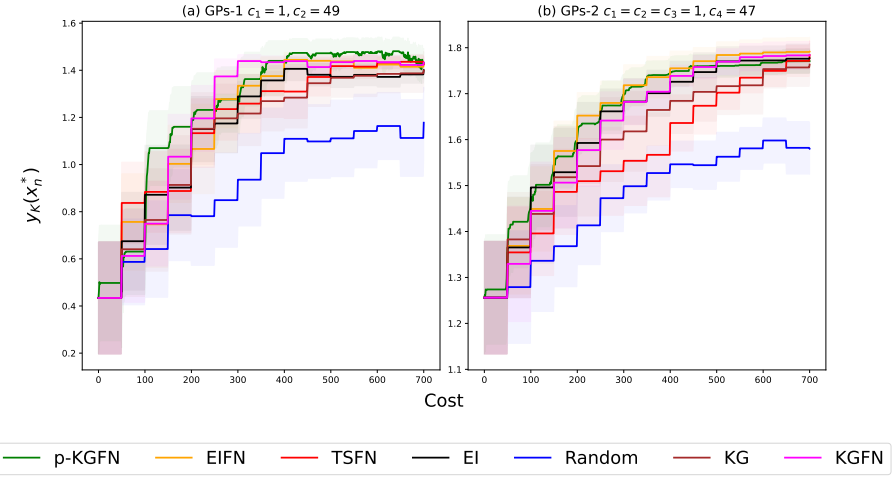

Figure 12: Optimization performance comparison between p-KGFN and benchmark algorithms on additional experiments where the first layer node is harder-to-learn than the second layer node. (a) GPs-1 with evaluation costs $c_1 = 1$, $c_2 = 49$ and (b) GPs-2 with evaluation costs $c_1 = c_2 = c_3 = 1$ and $c_4 = 47$. Both problems have BO budget equal to 700. Performance curves for p-KGFN and benchmarks, averaging over 30 replications. The mean and ± 2 standard errors of the mean over the evaluation budget used are reported.

We also conduct the sensitivity analysis study for an additional experiment: AckMat presented in Appendix D.5 for which we do not impose the upstream evaluation restriction.

Figure 13 reveals that performing partial evaluations notably improves optimization performance, especially when the costs of the two nodes are dramatically different. On the other hand, in the equal-cost scenario, p-KGFN takes less advantage of partial evaluating, tending to complete full evaluations in sequential networks (Ackley and FreeSolv), and chooses to evaluate the two properties in Pharm problem an equal number of times. Results for AckMat problem are presented in the last row of Figure 13 and are consistent with the previous three problems with upstream evaluation restriction. Table 2 reports the average number of times p-KGFN selected to evaluate each node in each problem and cost scenario.

Table 2: Number of times each node was evaluated by p-KGFN and benchmark algorithms (averaging over 30 replications) for different problems and costs of evaluation.

Problem	**Costs of evaluation**	**Average number of times each node was evaluated by p-KGFN**	**Number of full function network evaluations by benchmark algorithms**
Ackley	$c_1 = 1, c_2 = 1$	$[36.1, 13.9]$	25
Ackley	$c_1 = 1, c_2 = 9$	$[48.9, 11.2]$	15
Ackley	$c_1 = 1, c_2 = 49$	$[64.6, 13.0]$	14
Manufacturing	$c_1 = 5, c_2 = 10, c_3 = 10, c_4 = 45$	$[30.8, 13.9, 18.0, 5.0]$	10
FreeSolv	$c_1 = 1, c_2 = 1$	$[31.0, 19.0]$	25
FreeSolv	$c_1 = 1, c_2 = 9$	$[39.0, 12.3]$	15
FreeSolv	$c_1 = 1, c_2 = 49$	$[62.3, 12.9]$	14
Pharm	$c_1 = 1, c_2 = 1$	$[23.8, 26.0]$	25
Pharm	$c_1 = 1, c_2 = 9$	$[27.6, 13.6]$	15
Pharm	$c_1 = 1, c_2 = 49$	$[63.0, 13.0]$	14
AckMat	$c_1 = 1, c_2 = 1$	$[21.8, 28.2]$	25
AckMat	$c_1 = 1, c_2 = 9$	$[42.0, 12.0]$	15
AckMat	$c_1 = 1, c_2 = 49$	$[74.4, 12.8]$	14

F. Wall Clock Times

In this section, we report wall clock time on 8-core CPUs used to optimize each acquisition function on Ackley experiment.

Table 3: Acquisition optimization wall clock time in seconds on 8-core CPUs. Mean values and ± 2 standard errors are reported. KGFN takes significantly longer to optimize than p-KGFN because we use a larger number of samples when approximating its acquisition value.

Problem	**EI**	**KG**	**Random**	**EIFN**	**KGFN**	**TSFN**	**p-KGFN**
Ackley	6.7 ± 0.5	76.9 ± 4.5	0.00033 ± 0.00001	51.9 ± 4.8	1362.6 ± 50.1	7.7 ± 0.2	246.6 ± 5.1
Manufacturing	4.5 ± 1.1	54.4 ± 7.8	0.00025 ± 0.00001	29.5 ± 3.6	2047.8 ± 83.1	4.0 ± 0.1	302.6 ± 15.0
FreeSolv	3.4 ± 0.3	111.7 ± 10.8	0.00036 ± 0.00001	57.9 ± 5.3	1050.9 ± 98.3	1.8 ± 0.1	158.7 ± 6.0
Pharma	3.9 ± 0.3	27.6 ± 1.4	0.00033 ± 0.00001	14.6 ± 1.4	222.4 ± 13.7	5.9 ± 0.7	101.4 ± 3.2
AckMat	1.4 ± 0.1	306.7 ± 27.0	0.00029 ± 0.00001	40.4 ± 2.4	1634.5 ± 86.2	9.3 ± 0.3	508.0 ± 16.7

G. Additional Illustration of the Benefits of Partial Evaluations

In this section, we add the performance of KGFN with full evaluations to our 1-dimensional illustration example previously presented in Section 4.1 in order to highlight the substantial incremental benefits of performing partial evaluation. KGFN with full evaluations exhibits a similar behaviour to EIFN as depicted in the third row of Figure 14. Specifically, KGFN makes decisions towards its goal of identifying a point with the best solution quality. It first decides to evaluate around the initial best inferred solution and then spends two full evaluations exploring the boundaries of the domain where uncertainty is high. Focusing only on the final goal without taking evaluation costs into account makes KGFN fail to obtain an accurate final composite function model and a good best inferred solution (pink square).

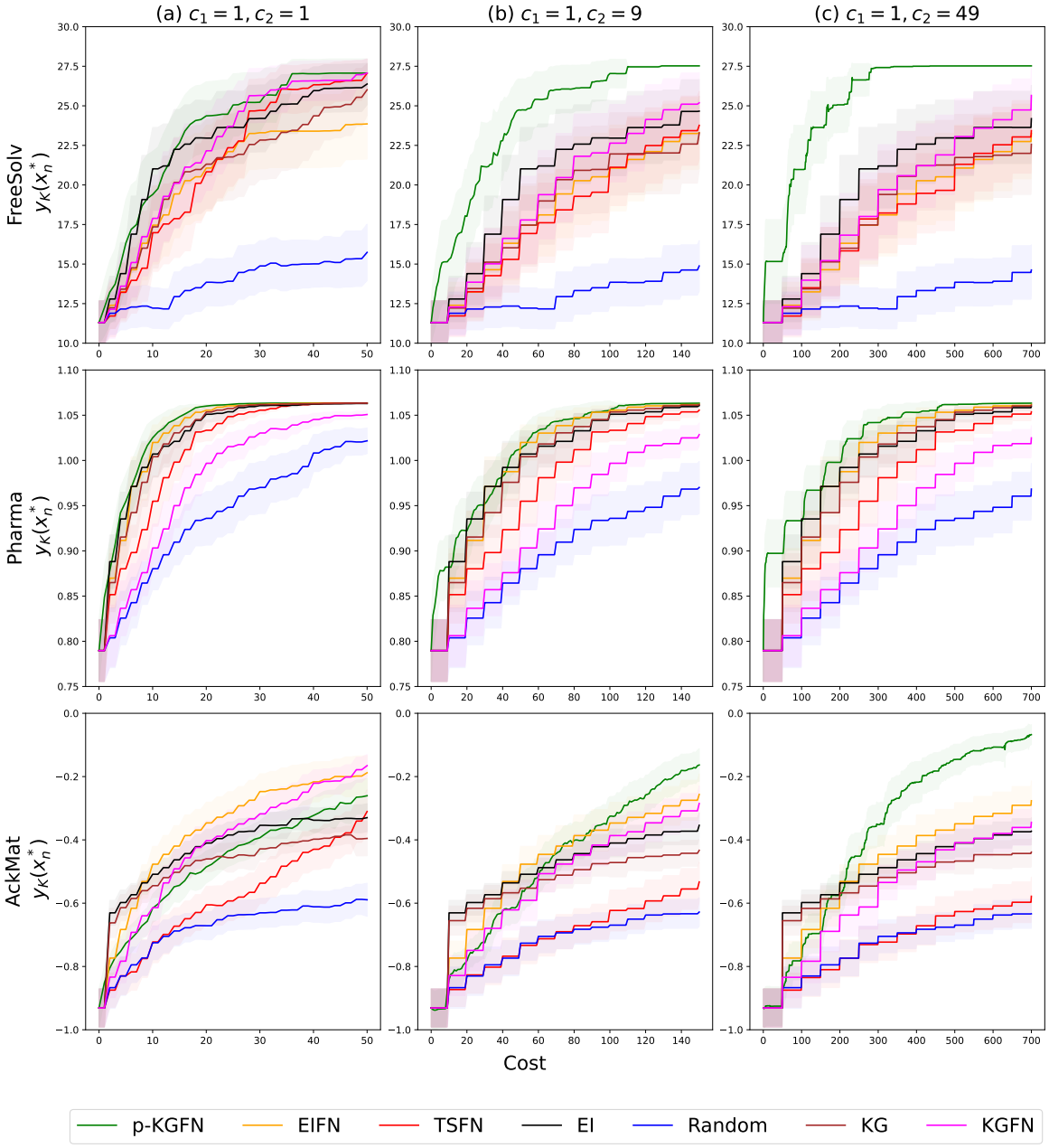

Figure 13: Cost sensitivity analysis for (from top to bottom) FreeSolv, Pharma and AckMat problems with different costs: (a) $c_1 = 1, c_2 = 1$; (b) $c_1 = 1, c_2 = 10$; and (c) $c_1 = 1, c_2 = 49$. The performance metric is the true objective value at the maximizer of final function node's posterior mean versus the budget spent.

H. Alternative Approach to Computing the Comparison Metric

As outlined in the main text, we employed a posterior distribution of the final node value y_K obtained from a statistical model that utilizes a network structure to compute our optimization comparison metric $y_K(x_n^*)$ across all algorithms including EI, KG and Random. The purpose is to underscore benefits of partial evaluations, but it unnecessarily favors these three algorithms as they do not actually consider a network structure in decision-making. In order to provide a more equitable comparison, we include the progress curves of the metric computed using a posterior distribution obtained from a standard Gaussian process model for these three algorithms. The results presented in Figure 15 illustrate, as expected, a degradation in their performance due to this modification. Notably, the Random baseline exhibits a declining trend in the AckMat problem when the network structure is not utilized. This is explained by the fact that the problem has a relatively small region of favorable outcomes.

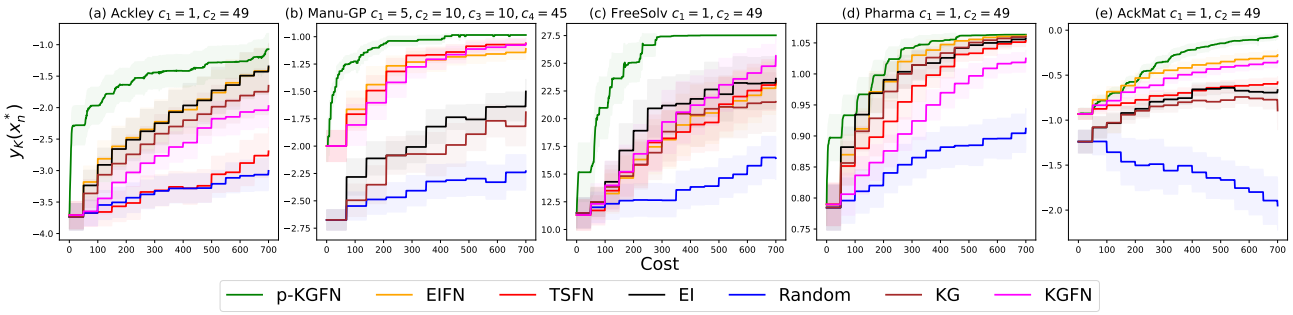

Figure 15: Optimization performance comparing between our proposed p-KGFN and benchmarks including EIFN, KGFN, TSFN, EI, KG and Random on four experiments: (a) Ackley, (b) Manu-GP, (c) FreeSolv, (d) Pharm and (e) AckMat. Every algorithm utilizes a statistical model in its decision-making process to calculate the comparison metric.

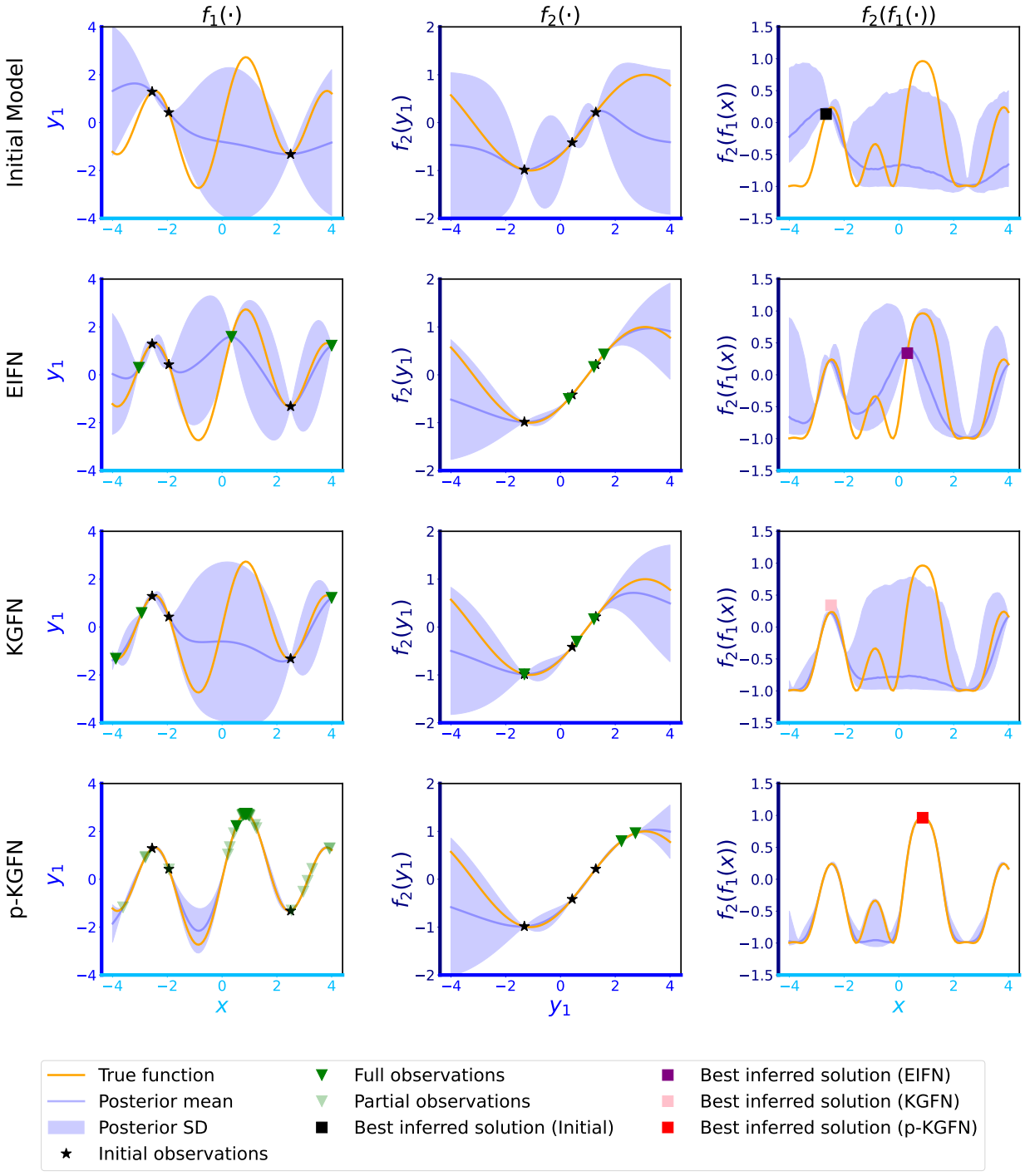

Figure 14: Comparison of EIFN, KGFN and p-KGFN on a 1-D synthetic two-stage function network $f_2(f_1(\cdot))$. The top row, from left to right, shows the initial models for $f_1(\cdot)$, $f_2(\cdot)$ and $f_2(f_1(\cdot))$. Similarly, the second, third and fourth rows show the resulting models upon budget depletion by EIFN, KGFN, and p-KGFN. Each true function is represented by an orange curve, while blue curves and shaded blue areas denote posterior mean functions and posterior uncertainty, respectively. Black stars indicate the initial three points fully evaluated across the network for both algorithms. Dark green triangles represent the locations of full network evaluations. Light green triangles represent partial observations where only the first node was evaluated by p-KGFN. Black, purple, pink and red squares correspond to the initial and three final best inferred solutions identified by EIFN KGFN, and p-KGFN, respectively.

References for the Supplementary Material

Ackley, D. *A connectionist machine for genetic hillclimbing*, volume 28. Springer science & business media, 2012.

Astudillo, R. and Frazier, P. Bayesian optimization of function networks. *Advances in Neural Information Processing Systems*, 34:14463–14475, 2021.

Astudillo, R., Jiang, D., Balandat, M., Bakshy, E., and Frazier, P. Multi-step budgeted Bayesian optimization with unknown evaluation costs. *Advances in Neural Information Processing Systems*, 34:20197–20209, 2021.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, 33:21524–21538, 2020.

Buckingham, J. M., Gonzalez, S. R., and Branke, J. Bayesian optimization of multiple objectives with different latencies. *arXiv preprint arXiv:2302.01310*, 2023.

Daulton, S., Balandat, M., and Bakshy, E. Hypervolume knowledge gradient: A lookahead approach for multi-objective Bayesian optimization with partial information. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 7167–7204, 2023.

Frazier, P., Powell, W., and Dayanik, S. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009.

Genton, M. G. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2 (Dec):299–312, 2001.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Jamil, M. and Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

Jiang, S., Jiang, D., Balandat, M., Karrer, B., Gardner, J., and Garnett, R. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33:18039–18049, 2020.

Mobley, D. L. and Guthrie, J. P. Freesolv: A database of experimental and calculated hydration free energies, with input files. *Journal of Computer-Aided Molecular Design*, 28:711–720, 2014.

Rubinstein, R. Y. and Shapiro, A. *Discrete event systems: sensitivity analysis and stochastic optimization by the score function method*, volume 13. Wiley, 1993.

Sano, S., Kadowaki, T., Tsuda, K., and Kimura, S. Application of Bayesian optimization for pharmaceutical product development. *Journal of Pharmaceutical Innovation*, 15:333–343, 2020.

Ungredda, J., Pearce, M., and Branke, J. Efficient computation of the knowledge gradient for Bayesian optimization. *arXiv preprint arXiv:2209.15367*, 2022.

Williams, C. K. and Rasmussen, C. E. *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA, 2006.