Bayesian Optimization of Function Networks

Raul Astudillo
Cornell University
ra598@cornell.edu

Peter I. Frazier
Cornell University
pf98@cornell.edu

Abstract

We consider Bayesian optimization of the output of a network of functions, where each function takes as input the output of its parent nodes, and where the network takes significant time to evaluate. Such problems arise, for example, in reinforcement learning, engineering design, and manufacturing. While the standard Bayesian optimization approach observes only the final output, our approach delivers greater query efficiency by leveraging information that the former ignores: intermediate output within the network. This is achieved by modeling the nodes of the network using Gaussian processes and choosing the points to evaluate using, as our acquisition function, the expected improvement computed with respect to the implied posterior on the objective. Although the non-Gaussian nature of this posterior prevents computing our acquisition function in closed form, we maximize it using a sample average approximation approach. In addition, we show that our method is asymptotically consistent, meaning that it finds a globally optimal solution as the number of evaluations grows to infinity, thus generalizing previously known convergence results for the expected improvement acquisition function. Notably, this holds even though it might not measure densely, instead leveraging problem structure to leave regions unexplored. Finally, our approach dramatically outperforms standard BO methods in several synthetic and real-world problems.

1 Introduction

We consider Bayesian optimization (BO) of objective functions defined by a series of time-consuming-to-evaluate functions, $f_1, \dots, f_K$, arranged in a directed acyclic network, so that each function takes as input the output of its parent nodes. As we detail below, these problems arise in BO-based policy search in reinforcement learning (Lizotte et al., 2007), optimization of complex systems modeled via simulation, and calibration of time-consuming physics-based models.

To illustrate, we introduce a running example of vaccine manufacturing (Sekhon and Saluja, 2011), focusing on the portion of the manufacturing process that uses live cells to produce proteins needed in a vaccine. It begins with a cell culture, in which living cells are grown and used as "factories" to produce proteins. This process is controlled by a vector, x_1, containing the temperature, pH, and CO_2 content used when growing these cells. The output of this process is the quantity of the desired protein $y_1 = f_1(x_1)$, i.e., the *yield* of this step, along with other byproducts. This output is passed into a second process, purification, which removes byproducts and is controlled by a vector x_2 comprising temperature, pressure, and flow rate. The yield of the desired protein from this second step is $y_2 = f_2(x_2, y_1)$. This output enters a third step, formulation, in which we formulate the raw protein into a form that can be distributed as controlled by a third set of parameters. This determines the yield of the overall process $y_3 = f_3(x_3, y_2)$. We wish to choose (x_1, x_2, x_3) to maximize overall protein yield. This problem is summarized as a function network in Figure 1.

The problem described above and other similar problems can be tackled with Bayesian optimization (BO), which has been shown to perform well compared to other derivative-free global optimization methods for time-consuming-to-evaluate objective functions (Snoek et al., 2012; Frazier, 2018). A

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

standard BO algorithm would fit a Gaussian process (GP) (Rasmussen and Williams, 2006) model on the objective function (y_3, which depends on (x_1, x_2, x_3)) and use it, along with an acquisition function, to sequentially choose the points to evaluate. Under this standard approach, however, evaluations of the intermediate nodes, $f_1, \ldots, f_{K-1}$, would be ignored despite being available when computing the objective function. In the example above, this corresponds to looking only at the yield of the overall process, and not of each individual step.

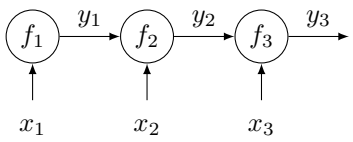

Figure 1: Vaccine manufacturing as a function network. Protein $y_1 = f_1(x_1)$ is created, then purified with yield $y_2 = f_2(x_2, y_1)$, and formulated with yield $y_3 = f_3(x_3, y_2)$. The goal is to find (x_1, x_2, x_3) that maximizes y_3.

In this paper, we introduce a novel BO approach that leverages function network structure for substantially more efficient optimization. This approach models the individual nodes of the network using distinct GPs. This allows incorporating observations of each node's output recursively into a non-Gaussian posterior on the network's overall output. Our approach then chooses the points to evaluate using the expected improvement (Jones et al., 1998) computed with respect to this implied posterior on the objective function. The non-Gaussian nature of this posterior prevents the expected improvement from having a closed form. However, we show that it can still be efficiently maximized via sample average approximation (Kim et al., 2015).

Our approach can outperform standard BO by leveraging information from internal nodes unavailable to standard methods. We briefly explain one way this can happen, in the context of the example above. In vaccine manufacturing, each function $f_k(x_k, y_{k-1}),\ k = 2, 3$, is bounded between 0 and y_{k-1} because new protein cannot be created in the purification and formulation stages. Moreover, application experts have a prior on what values for $f_k(x_k, y_{k-1})/y_{k-1}$ should be achievable if x_k is set well. Thus, if we see that y_3 is unexpectedly poor, information from intermediate nodes can be extremely valuable: if y_1 and y_2 are as expected, then this suggests the problem is with x_3; if y_1 is as expected but y_2 is poor, then the problem is likely with x_2; and if y_1 is poor then the problem is likely with x_1. (If there is a problem with x_k, there may also be a problem with $x_{k'}$, $k' > k$, but we can focus on fixing x_k first.) Thus, by observing intermediate nodes, we can instantly reduce the effective dimensionality of the input space by a factor of $K = 3$.

We show that our method is consistent, i.e., that it discovers the global optimum given sufficiently many samples. In contrast with most BO methods, it may do so without measuring densely in the feasible domain, instead leveraging function network structure to exclude regions as unnecessary to explore. This indicates the power of function network structure to improve query efficiency.

We demonstrate through numerical experiments that access to additional information available in a problem formulated as a function network can dramatically accelerate optimization. We study four synthetic problems and four real-world problems: a manufacturing problem similar in spirit to the vaccine example above, active learning for robot pushing, and two problems arising in epidemiology, one calibrating an epidemic model and the other designing a testing strategy to control the spread of COVID-19. Our method significantly outperforms competing methods that utilize less information, in some cases by $\sim$5% and in other cases by several orders of magnitude.

2 Related Work

Our work occurs within BO, a framework for global optimization of expensive-to-evaluate black-box functions that originated with the work of Zhilinskas (1975) and Močkus (1975), and has recently become popular due to its remarkable performance in hyperparameter tuning of machine learning algorithms (Snoek et al., 2012; Swersky et al., 2013; Wu et al., 2019). See Shahriari et al. (2016) and Frazier (2018) for modern introductions to BO.

Our approach can be catalogued as a *grey-box* BO method since it does not treat the objective function entirely as a black box, and instead exploits known structure to improve sampling efficiency. Other

examples of grey-box BO approaches include multi-fidelity BO (Kandasamy et al., 2017; Poloczek et al., 2017), which leverages cheaper approximations of the objective function; BO of objective functions that are the integral of an expensive-to-evaluate integrand (Williams et al., 2000; Toscano-Palmerin and Frazier, 2018); BO of objective functions that are a sum of squared errors (Uhrenholt and Jensen, 2019); and, more generally, BO of objective functions that are the composition of of an expensive-to-evaluate inner function and a known inexpensive-to-evaluate outer function (Astudillo and Frazier, 2019), of which our work can be seen as a significant generalization.

Our work is also closely related to Marque-Pucheu et al. (2019), which proposes a method for efficient sequential experimental design of nested computer codes, also using GPs. This work focuses on the case where there are only two node functions, and one takes as input the output of the other. In contrast with our work, the goal of the proposed method is to learn the output code as accurately as possible within a limited evaluation budget, but optimization is not pursued.

Optimization of composite (a.k.a nested) functions has also been considered in the gradient-based optimization literature (Shapiro, 2003; Drusvyatskiy and Paquette, 2019; Charisopoulos et al., 2019; Balasubramanian et al., 2020). In contrast with ours, these works assume that evaluations are inexpensive, and often also some form of convexity, along with availability of gradients. However, this literature is similar in spirit to ours in that information from inner functions improves efficiency.

Function networks arise in many application areas. While these applications have not, to our knowledge, been previously formulated as specific instances of the general function network model we propose, their literatures are nonetheless relevant to our work.

- In engineering and aerospace design, function networks arise in multidisciplinary optimization (Cramer et al., 1994; Amaral et al., 2014; Benaouali and Kachel, 2019), where simulators focusing on different physical laws are coupled into a function network. For example, a simulation of airflow over a wing may output the forces on the wing to another simulation of mechanical stress on the wing structure.
- In drug discovery and materials design, function networks arise from the sequence-structure-function Sadowski and Jones (2009) and composition-structure-property (Hattrick-Simpers et al., 2016) paradigms. Here, decision variables (composition, e.g., what fraction of what raw materials are used) determine the structure of the material (how the atoms combine together), which in turn determines how the material behaves (properties).
- Function networks arise in the design of queuing networks (Fu and Henderson, 2017; Arcelli, 2020). This includes manufacturing systems (Ghasemi et al., 2018), where the partially-processed output of one workstation is input to the next workstation, the design of service systems (Wang et al., 2020) such as hospitals and airport security checkpoints, and choosing traffic signal timings for a city's street network (Osorio and Bierlaire, 2013).
- Finally, function networks arise in reinforcement learning (Sutton and Barto, 2018) and Markov decision processes (Puterman, 1990), where the transition kernel transforms the state variable at the start of a time range to another state variable at the end of this range. This outputted state variable is the input to the next time range. §5.2 shows an example.

3 Problem Setting

We consider objective functions evaluated via a series of functions, $f_1, \ldots, f_K$, arranged in a directed network so that each function in this network takes as input the output of its parent nodes, and assume that evaluating this collection of functions is time-consuming. Formally, we encode this network structure as a directed graph with nodes $V = \{1, \ldots, K\}$ and directed edges $E = \{(j, k) : f_k \text{ take as input the output of } f_j\}$. We assume that this graph is acyclic and has a single leaf node whose output is the objective to optimize.

Let $J(k)$ denote the set of parent nodes of node k[1]. We assume, without loss of generality, that the nodes are ordered such that $j < k$ for all $j \in J(k)$. In addition to the output of its parent nodes, we assume that each function, f_k, takes as input a (potentially empty) subset, $I(k) \subset \{1, \ldots, D\}$, of the components of the decision vector $x \in \mathbb{R}^D$.

[1] We allow $J(k) = \emptyset$, which corresponds to the root node(s).

Let $h_1(x), \ldots, h_K(x)$ denote the values of the K nodes in the function network when it is evaluated at x. These values are defined recursively by

$$h_k(x) = f_k\left(x_{I(k)}, h_{J(k)}(x)\right),\ k = 1, \ldots, K,$$

i.e., by evaluating each function in the network as the values of its parent nodes become available. This recursion is well-defined by our assumption that the graph is acyclic. The objective function is then $g = h_K$, and the goal is to solve

$$\max_{x \in \mathbb{X}} g(x),$$

where $\mathbb{X} \subset \mathbb{R}^D$ is a simple compact set, such as a hyper-rectangle.

The standard BO approach to this problem models g using a GP prior distribution. This approach iteratively chooses the next point at which to evaluate g as follows. Given n observations of the objective function, $g(x_1), \ldots, g(x_n)$, it computes the posterior distribution on g, which is itself a GP. It then uses this posterior distribution to compute an acquisition function (Frazier, 2018) that quantifies the value of the information that would result from a function evaluation at any given point. Finally, it chooses the next point to evaluate, x_{n+1}, as the point that maximizes this acquisition function. Notably, although the outputs of $f_1, \ldots, f_{K-1}$ would be observed as part of this approach, these evaluations would be ignored by standard BO when calculating the posterior distribution and corresponding acquisition function.

4 Bayesian Optimization with Full Network Observations

This section describes our approach. Like standard BO, it consists of a statistical model and an acquisition function. Unlike standard BO, however, our approach leverages the network structure of the problem by utilizing intermediate outputs within the network. As we describe below, this is achieved by modeling the node functions, $f_1, \ldots, f_K$, as GPs, which in turn implies a non-Gaussian distribution on g (§4.1). Our acquisition function is the expected improvement under this posterior distribution (§4.2), which no longer has a closed form and thus we maximize it via sample average approximation (§4.3). We end up this section by proving that our acquisition function is asymptotically consistent despite not necessarily sampling the feasible space densely (§4.4).

4.1 Statistical Model

Instead of modeling g directly, we model $f_1, \ldots, f_K$, as drawn from independent GP prior distributions with mean and covariance functions $\mu_{0,k}$ and $\Sigma_{0,k}$, $k = 1, \ldots, K$. Importantly, when g is evaluated at x, we also get to observe the value of f_k at $\left(x_{I(k)}, h_{J(k)}(x)\right)$. Therefore, after querying g at n points, $x_\ell,\ \ell = 1, \ldots, n$, the set of observations of the values of f_k at $\left(x_{\ell,I(k)}, h_{J(k)}(x_\ell)\right)$, $\ell = 1, \ldots, n$, imply posterior distributions on $f_1, \ldots, f_K$, which are again (conditionally) independent GPs with mean and covariance functions $\mu_{n,k}$ and $\Sigma_{n,k}$[2], $k = 1, \ldots, K$. These can be computed in closed form using the standard GP regression equations (see, e.g., Rasmussen and Williams 2006). For completeness, we include these equations in the supplement.

The posterior distributions on $f_1, \ldots, f_K$, described above imply a posterior distribution on g. This distribution is in general non-Gaussian and thus, unlike in the standard setting, where g is modeled as a GP, classical acquisition functions such as the expected improvement cannot be computed in closed form. However, as we describe next, drawing samples from this distribution is simple.

Thanks to the acyclic structure of the underlying network that defines g, a sample, $\widehat{g}(x) = \widehat{h}_K(x)$ from the posterior distribution on g at x can be obtained following the iterative process described next. In each iteration, k, we obtain a sample, $\widehat{h}_k(x)$, from the posterior distribution on $h_k(x)$ by drawing a sample from the normal distribution with mean $\mu_{n,k}\left(x_{I(k)}, \widehat{h}_{J(k)}(x)\right)$ and standard deviation

$$\sigma_{n,k}\left(x_{I(k)}, \widehat{h}_{J(k)}\right) = \Sigma_{n,k}\left(x_{I(k)}, \widehat{h}_{J(k)}(x), x_{I(k)}, \widehat{h}_{J(k)}(x)\right)^{1/2}.$$

[2]For ease of presentation we assume that all the functions in the network are expensive-to-evaluate, and thus require to be modeled as GPs. However, if any of the functions, say f_k, is not expensive-to-evaluate, we can simply take $\mu_k \equiv f_k$ and $\Sigma_k \equiv 0$.

Supporting efficient calculation, the samples $\widehat{h}_{J(k)}(x)$ will have already been obtained in previous iterations of the for loop since $j < k$ for all $j \in J(k)$ (Note that $J(1) = \emptyset$). This procedure is summarized in Algorithm 1.

Algorithm 1 Draw one sample from the posterior on $g(x)$

Require: $x \in \mathbb{X}$
1: **for** $k = 1, \ldots, K$ **do**
2: $\quad \widehat{h}_k(x) \sim \mathcal{N}\left(\mu_{n,k}\left(x_{I(k)}, \widehat{h}_{J(k)}(x)\right), \sigma_{n,k}\left(x_{I(k)}, \widehat{h}_{J(k)}(x)\right)^2\right)$
3: **end for**
4: **return** $\widehat{g}(x) = \widehat{h}_K(x)$

We end this section by noting that, while the statistical model described above could be catalogued as a deep GP (Damianou and Lawrence, 2013), in the sense that we have GP layers in an architecture described by a directed acyclic network, inference in our model is faster. Typically, deep GP inference requires marginalization over latent values of GP layers. In our setting, however, observation structure creates conditional independence across layers, avoiding the need to marginalize.

4.2 Expected Improvement for Function Networks

Our acquisition function is the expected improvement, computed with respect to the implied posterior distribution on g under the statistical model described in §4.1:

$$\text{EI-FN}_n(x) = \mathbb{E}_n\left[\{g(x) - g_n^*\}^+\right],$$

where $g_n^* = \max_{i=1,\ldots,n} g(x_n)$ is the best value observed so far, and $\mathbb{E}_n$ is the expectation computed with respect to the GP time-n posterior distributions on $f_1, \ldots, f_K$. To distinguish it from the classical expected improvement, we refer to our acquisition function as the expected improvement for function networks (EI-FN) in the remainder of this work.

4.3 Maximization of EI-FN via Sample Average Approximation

Since the posterior distribution on g is non-Gaussian, in contrast with the classical expected improvement acquisition function, EI-FN$_n$ does not admit a closed form. However, EI-FN$_n(x)$ can be computed with arbitrary precision following a simple Monte Carlo (MC) approach:

$$\text{EI-FN}_n(x) \approx \frac{1}{M}\sum_{m=1}^{M}\left\{\widehat{g}(x)^{(m)} - g_n^*\right\}^+,$$

where $\widehat{g}(x)^{(1)}, \ldots, \widehat{g}(x)^{(M)}$ are samples drawn from the posterior distribution on $g(x)$, which can be obtained following the approach described in §4.1.

Following the above MC scheme to approximately compute EI-FN, one can aim to maximize this function using a derivative-free global optimization algorithm for inexpensive-to-evaluate functions, such as CMA (Hansen, 2016), by drawing samples, $\widehat{g}(x)^{(1)}, \ldots, \widehat{g}(x)^{(M)}$ independently for each x at which EI-FN$_n$ is evaluated. However, this approach is slow since evaluations of EI-FN are noisy and derivative information is not leveraged. Instead, we propose to maximize EI-FN following a sample average approximation (SAA) approach (Kim et al., 2015).

Succinctly, the SAA approach works by building a MC approximation of EI-FN that is deterministic given a finite set of random variables not depending on x, and maximizing this approximation instead of EI-FN itself. The key observation for creating this approximation is that a sample, $\widehat{g}(x)$, can be obtained as $\widehat{g}(x) = \widehat{h}_K(x)$, where $\widehat{h}_1(x), \ldots, \widehat{h}_K(x)$ are defined recursively by

$$\widehat{h}_k(x; Z) = \mu_{n,k}\left(x_{I(k)}, \widehat{h}_{J(k)}(x; Z)\right) + \sigma_{n,k}\left(x_{I(k)}, \widehat{h}_{J(k)}(x; Z)\right) Z_k,$$

where $Z = (Z_1, \ldots, Z_K)^\top$ is a standard normal random vector, and we write $\widehat{h}_k(x)$ as $\widehat{h}_k(x; Z)$ to make its dependence on Z explicit. Analogously, we also write $\widehat{g}(x)$ as $\widehat{g}(x; Z)$. This can be seen as an extension of the so-called reparametrization trick for acquisition functions (Wilson et al., 2018).

If we now fix M samples drawn from the K-dimensional standard normal distribution, $Z^{(1)}, \ldots, Z^{(M)}$, then

$$\widehat{\text{EI-FN}}_n\left(x; Z^{(1:M)}\right) := \frac{1}{M}\sum_{m=1}^{M}\left\{\widehat{g}\left(x; Z^{(m)}\right) - g_n^*\right\}^+$$

is a MC approximation of EI-FN that is deterministic given $Z^{(1)}, \ldots, Z^{(M)}$, as desired. Moreover, Proposition 1 below shows that, under mild regularity conditions, any maximizer of the above SAA converges in probability exponentially fast to a maximizer of EI-FN$_n$ as $M \to \infty$, thus suggesting that in practice it is safe to use low values of M. This result is a simple generalization of Theorem 1 in Balandat et al. (2020). Its proof can be found in the supplement.

Proposition 1. *Suppose that the functions $\mu_{n,k}$ and $\sigma_{n,k}$, for $k = 1, \ldots, K$, are all Lipschitz continuous and let*

$$\widehat{x}_*^{(M)} \in \operatorname*{argmax}_{x\in\mathbb{X}} \widehat{\text{EI-FN}}_n\left(x; Z^{(1:M)}\right), \quad X_* = \operatorname*{argmax}_{x\in\mathbb{X}} \text{EI-FN}_n(x).$$

Then, for every $\epsilon > 0$, there exist $A, \alpha > 0$ such that $\mathbb{P}\left(\text{dist}\left(\widehat{x}_*^{(M)}, X_*\right) > \epsilon\right) \leq Ae^{-\alpha M},\ M \in \mathbb{N}$.

Finally, we note that $\widehat{\text{EI-FN}}_n$ is differentiable with respect to x, provided that $\mu_{k,n}$ and $\Sigma_{k,n}$, $k = 1, \ldots, K$ are all differentiable. Thus, $\widehat{\text{EI-FN}}_n$ can be maximized efficiently using a gradient-based deterministic optimization algorithm such as L-BFGS (Liu and Nocedal, 1989), with multiple restarts.

4.4 Asymptotic Consistency of EI-FN without Dense Measurements

To shed light on the convergence properties of the expected improvement acquisition function under our statistical model, we prove that, under suitable regularity conditions, EI-FN is asymptotically consistent, meaning that it finds the global optimum of the objective function as the number of evaluations grows to infinity. This builds on results shown for the classical expected improvement (Vazquez and Bect, 2010; Bect et al., 2019) and the expected improvement for composite functions by and (Astudillo and Frazier, 2019), and we rely on similar assumptions. This result is stated in Theorem 1 below and its proof can be found in the supplement.

Theorem 1. *Suppose that $x_{n+1} \in \operatorname{argmax}_{x\in\mathbb{X}} \text{EI-FN}_n(x)$ for all n. Then, under regularity conditions stated precisely in the supplement, $g_n^* \to \max_{x\in\mathbb{X}} g(x)$ as $n \to \infty$ almost surely under the prior distribution on $f_1, \ldots, f_K$.*

Significant novelty in our proof arises from the fact that EI-FN's measurements are not necessarily dense in $\mathbb{X}$. In nearly all existing work, consistency of BO methods is shown by first showing that the measurements are dense in the objective function's domain (e.g., Vazquez and Bect 2010). Surprisingly, the function network structure allows us to exclude certain regions of $\mathbb{X}$ as suboptimal after only finitely many measurements, allowing our method to be consistent without measuring everywhere. Such ability gives insight into EI-FN's strong empirical performance compared to methods ignoring function network structure. As stated in Proposition 2, we provide a function network where EI-FN provides a consistent estimate of the global optimum *without* sampling densely.

Proposition 2. *There exists a function network (detailed in the supplement) in which EI-FN is consistent but whose measurements are not necessarily dense in $\mathbb{X}$.*

5 Numerical Experiments

We compare the performance of our algorithm (EI-FN) against the classical expected improvement (EI), i.e., the expected improvement under a GP model over g. We also compare with the performance of two other standard algorithms: the algorithm that chooses the points to evaluate uniformly at random over $\mathbb{X}$ (Random); and the knowledge gradient (KG) (also under a GP model over g), a more sophisticated acquisition function that often delivers a better performance than the expected improvement (Wu and Frazier, 2016). The problems in §5.2 and §5.3 fall within the framework of Astudillo and Frazier (2019) and we include its proposed EI-CF algorithm as a benchmark. All algorithms were implemented in BoTorch (Balandat et al., 2020).

We evaluate on 4 synthetic problems and 4 real-world problems. These are described below or in the supplement, with the supplement describing a manufacturing problem building on §1 and a COVID-19 testing problem building on §5.3. In all problems, a first stage of evaluations is performed using $2(d+1)$ points chosen uniformly at random over $\mathbb{X}$. A second stage (pictured in plots) is then performed using each of the algorithms. Error bars show the mean of either the best objective value observed so far or the base-10 logarithm of the optimality gap, plus and minus 1.96 times the standard deviation divided by the square root of the number of replications. Each experiment was replicated 30 times. Experimental setup details and runtimes are available in the supplement. Code to reproduce our numerical experiments can be found at `https://github.com/RaulAstudillo06/BOFN`.

5.1 Synthetic Test Functions

We create synthetic test problems by arranging standard test functions from the global optimization literature (Jamil and Yang, 2013) into function networks. These explore a variety of network structures in an easy-to-reproduce form, and are named after the standard test function used to define the function network. We describe these briefly here and then in full detail in the supplement.

Alpine2 and **Rosenbrock** both arrange K nodes in series, where each node except the first node takes the output of the previous node as input. Additionally, in Alpine2, each node takes a distinct dimension of the decision vector x as input. In Rosenbrock, x_1 and x_2 are inputs to the first node, x_2 and x_3 are inputs to the second, and so on. These network architectures arise in manufacturing problems like the example in §1, as well as business operations with queues like boarding an aircraft or fulfilling drive-through orders. We consider $K = 2, 4, 6$ nodes. The plots for the cases $K = 2, 4$, and $K = 2, 6$ for Alpine2 and Rosenbrock, respectively, are deferred to the supplement.

Ackley has 3 nodes. The first two nodes each take the same 6-dimensional input. Their outputs are passed to the third node that produces the final output. This type of function network arises in algorithm design for two-sided markets (Li et al., 2021), like Uber and AirBnB, where the first node simulates an intervention's effect on riders (or guests), the second simulates its effect on drivers (or hosts), and the third simulates the matching process where riders and drivers (or guests and hosts) interact to produce transactions.

Drop-Wave has two nodes. The first node takes a two-dimensional vector x as input. This node's output is passed to the second node, which produces the objective value. This network architecture is representative of multidisciplinary engineering design (Benaouali and Kachel, 2019), for example in aerospace, where a small number of distinct black-box simulators simulate processes governed by physical laws that affect each other through a small number of channels, such as an aircraft engine simulation (the first node) determining heat generated while flying, which is then inputted to a temperature-dependent simulation of mechanical stress on the aircraft's frame (the second node).

5.2 Active Learning for Robot Pushing

We consider a variation of the active learning for robot pushing problem introduced by Wang and Jegelka (2017) whose goal is to teach a robot to push an object to a predetermined target location. We modify the problem by allowing the robot to push the object several times instead of only once. We formalize this problem as follows. Let $x_{\text{init}}, x_{\text{target}} \in [-5, 5]^2$ denote the object's initial and target locations, respectively. At each time step, t, we choose the location of the robot's arm, $r_t \in [-5, 5]^2$, and the duration of the push, $d_t \in [1, 12]$. The robot then moves its arm from r_t in the current direction of the object, x_t, over d_t units of time. After this push, the location of the object becomes x_{t+1} (if the robot fails to push the object, $x_{t+1} = x_t$). The goal is to choose (r_t, d_t) for $t = 1, \ldots, T$ to minimize $\|x_{\text{target}} - x_T\|_2^2$. We set $x_{\text{init}} = (0, 0)$, $x_{\text{target}} = (2.9, 1.6)$, and $T = 3$. This can be interpreted as a function network by associating each time step with a pair of node functions $p_{t,1}$ and $p_{t,1}$ which take (r_t, d_t, x_t) as input and produce $x_{t+1} = (p_{t,1}(r_t, d_t, x_t), p_{t,2}(r_t, d_t, x_t))$ as output.

A very similar experiment to the one described above can be found in Section 5 of the supplementary material. It considers a fetch-and-reach path planning problem using Open AI Gym. As in the experiments here, EI-FN outperforms other methods, including EI-CF.

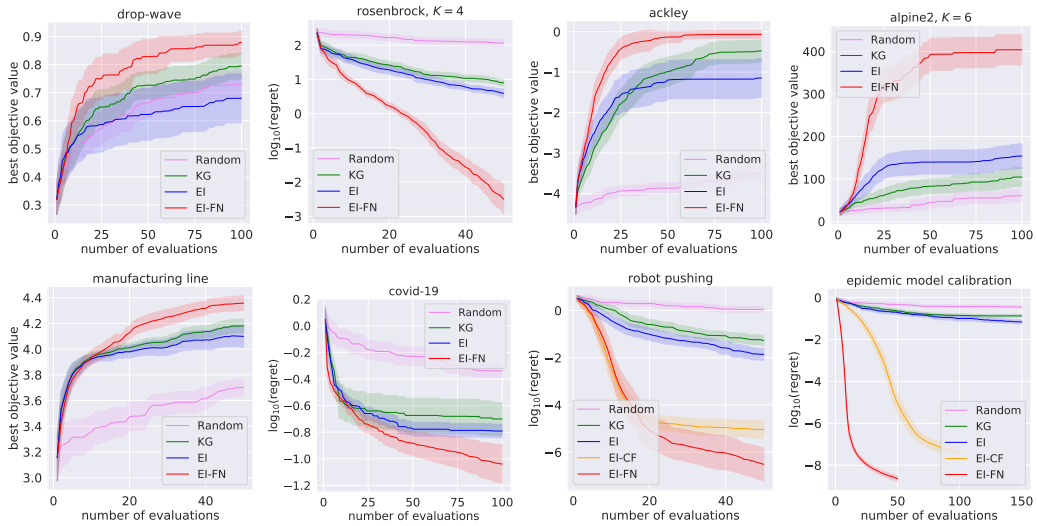

Figure 2: Top: Results on synthetic problems that adapt widely used synthetic test functions into function networks. Bottom: Results on realistic problems: manufacturing line, design of testing protocols for COVID-19, active learning for robot pushing, and calibration of an epidemic model. EI-FN substantially improves over benchmark methods, with larger improvements for problems with higher-dimensional decision vectors and more nodes.

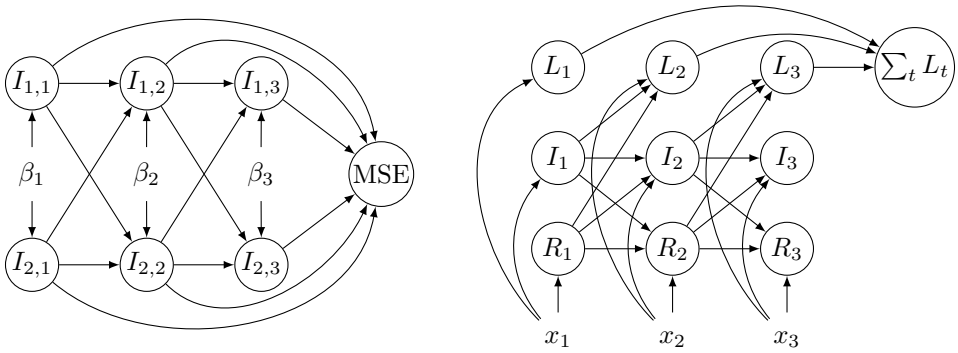

Figure 3: Function network for the (left) epidemic calibration problem in §5.3 and (right) the COVID-19 pooled testing optimization problem described in the supplement.

5.3 Calibration of an Epidemic Model

Here we consider calibration of compartmental stochastic models to data, a widely-used tool in epidemiology, medical modeling, and ecology (Sandberg, 1978). Function networks are well-suited to exploit the structure of such models. We focus on calibration of a specific epidemic model for influenza, building toward a COVID-19 mitigation benchmark in the next section. We first describe the epidemic model, then the calibration problem and, finally, formulation as a function network.

SIS Epidemiological Model: We calibrate to data a widely used epidemiological model, the SIS model (see, e.g., Garnett 2002), that models diseases like influenza capable of reinfecting individuals multiple times. In this model, individuals either do not have the disease and are "susceptible" (S) or have the disease and are "infectious" (I). We consider a SIS model of two interacting populations, where infections occur at population-dependent rates.

The model is dynamic, with time indexed by $t = 0, 1 \ldots, T$. At the beginning of each time period t, the fraction of population $i \in \{0, 1\}$ that is infectious is $I_{i,t}$. We assume each population is of equal size, N. During this time period, each person in group i comes into close physical contact with $\beta_{i,j,t}$

people from group j. When this contact is between an infectious person and a susceptible one, it infects the susceptible person. A fraction $I_{j,t}$ of the people from group j involved in such interactions are infectious and a fraction $(1 - I_{i,t})$ from group i are susceptible. A number of new infections in group i result, $N(1 - I_{i,t})\beta_{i,j,t}I_{j,t}$. As a fraction of group i's population, this is $(1 - I_{i,t})\beta_{i,j,t}I_{j,t}$. Summing across j, we have $(1 - I_{i,t})\sum_j \beta_{i,j}(t)I_{j,t}$ new group i infections. At the same time, infections resolve at a rate of γ per period. This results in a decrease in the infectious population in group i of $\gamma I_{i,t}$. Putting this together, the number of infectious individuals in group i at the start of the next time period is $I_{i,t+1} = I_{i,t}(1 - \gamma) + (1 - I_{i,t})\sum_j \beta_{i,j,t}I_{j,t}$.

Calibration: The SIS model has parameters $I_{i,0}$, γ and $\beta_{i,j,t}$, where $0 \leq t < T$ and $i, j \in \{1, 2\}$. We calibrate the parameters $\vec{\beta} = (\beta_{i,j,t} : i, j, t)$ while fixing $\gamma = 0.5$ and $I_{i,0} = .01$ (for both i). We simulate a trajectory of infections from $t = 0$ up to $T = 3$, using a held-out value for $\vec{\beta}$. We let $I^{\text{obs}}_{i,t}$ denote the fraction of group i observed to be infected at time t in this trajectory. We then search for the vector $\vec{\beta}$ that, when passed to the SIS model, minimizes the mean squared error between this trajectory and the SIS model predictions. Letting $I_{i,t}(\vec{\beta})$ indicate this predicted value, the goal is to minimize the mean-squared error (MSE), $\text{MSE}(\vec{\beta}) := \sum_{t=1}^{T}(I^{\text{obs}}_{i,t} - I_{i,t}(\vec{\beta}))^2$. We do not include $t = 0$ since $I_{i,0}(\vec{\beta})$ is the same for all $\vec{\beta}$.

Formulation as a Function Network: We encode this as a function network using $2T + 1$ nodes, as illustrated in Figure 3. For each time period t and each group i, a node takes input $I_t := (I_{j,t} : j = 0, 1)$ and $\beta_t := (\beta_{j,j',t} : j, j' \in \{0, 1\})$ and produces output $I_{i,t+1}$. (For $t = 0$, the input I_0 is not needed since this is the same for all $\vec{\beta}$.) Then, one additional node takes the output of the other nodes $(I_{i,t} : i = 0, 1, t = 1, \ldots, T)$ as its input and produces the sum of squared errors $\sum_{t=1}^{T}(I^{\text{obs}}_{i,t} - I_{i,t}(\vec{\beta}))^2$ as output. We treat this final node as known (its GP prior has a kernel of 0).

EI-CF benchmark: The fact that the final node in this problem (denoted "MSE" in Figure 3) has known structurre permits comparing against the EI-CF method for BO of composite functions (Astudillo and Frazier, 2019) as a benchmark. EI-CF is substantially less general than our method (EI-FN): it is restricted to settings with one time-consuming black-box multi-output node that provides input to one fast-to-evaluate node with known structure. To apply EI-CF to this problem, the single black-box multi-output node takes $\vec{\beta}$ as input and produces the vector $(I_{i,t}(\vec{\beta}) : i, t)$ as output. This output is then supplied to the "MSE" node. This approach ignores the fact that $I_{i,t}$ does not depend on $\beta_{t'}$, $t' > t$ and depends only indirectly on $\beta_{t'}$, $t < t$ through $I_{j,t-1}$, $j = 1, 2$.

5.4 Results and Discussion

Figure 2 presents results of our experiments, with additional results in the supplement. Across the wide range of problems considered, EI-FN significantly improves query efficiency over standard BO methods that ignore the function network structure of evaluations. The benefits range from a 5% improvement in the value of the best point found on the Drop-Wave and manufacturing problems to several orders of magnitude in the Rosenbrock and epidemic model calibration problems.

The largest benefit arise when the control input is high-dimensional but the input to individual nodes is low-dimensional. On the epidemic model calibration problem, we see EI-CF (Astudillo and Frazier, 2019) outperforming EI and KG by several orders of magnitude, and EI-FN outperforming EI-CF by several *additional* orders of magnitude. As noted above, EI-CF can be seen as a special case of EI-FN using a less informative function network that hides observations from some nodes. This is consistent with observations of function network structure allowing substantial improvement in query efficiency, and observing more of the internal function network structure providing more value.

6 Conclusion

We introduced a novel BO approach for objective functions defined by a series of expensive-to-evaluate functions, arranged in a network so that each function takes as input the output of its parent nodes. These objective functions arise in a wide range of settings where BO has been applied. However, existing methods cannot leverage this structure. Our approach models the outputs of the functions in this network instead of only the objective function, as is standard in BO. Our experiments show that, by doing so, this approach can dramatically outperform standard BO methods.

Though we see substantial benefits from our approach, there are limitations. First, our approach requires more computation than standard BO methods, as explored in the supplement. (When the objective is time-consuming, the improved query efficiency more than makes up for the additional computation required.) Second, while we have demonstrated our method in problems with up to 9 nodes, and computational speed would support more, our method does not (yet) scale to hundreds of nodes. Third, while we show consistency, it would be instructive to complement our empirical results showing fast convergence with a theoretical understanding of convergence rates. Existing approaches for BO convergence rates for the classical expected improvement heavily rely on properties of its analytical expression (Bull, 2011; Ryzhov, 2016), and thus are not directly generalizable to our setting. This is, however, an exciting direction for future work.

As with any powerful new method for optimizing time-consuming-to-compute black-box functions, ours can accelerate many applications. While this includes innovations that generally benefit society, such as improvements to public health and vaccine manufacturing, it also includes the design of weapons and other engineering systems that could harm individuals. Thus, it is important that society enact guardrails that ensure proper use of our methodology.

Acknowledgments

The authors were partially supported by AFOSR FA9550-19-1-0283 and FA9550-20-1-0351.

References

Amaral, S., Allaire, D., and Willcox, K. (2014). A decomposition-based approach to uncertainty analysis of feed-forward multicomponent systems. *International Journal for Numerical Methods in Engineering*, 100(13):982–1005.

Arcelli, D. (2020). Exploiting queuing networks to model and assess the performance of self-adaptive software systems: a survey. *Procedia Computer Science*, 170:498–505.

Astudillo, R. and Frazier, P. (2019). Bayesian optimization of composite functions. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 354–363, Long Beach, California, USA. PMLR.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*.

Balasubramanian, K., Ghadimi, S., and Nguyen, A. (2020). Stochastic multi-level composition optimization algorithms with level-independent convergence rates. *arXiv preprint arXiv:2008.10526*.

Bect, J., Bachoc, F., and Ginsbourger, D. (2019). A supermartingale approach to gaussian process based sequential design of experiments. *Bernoulli*, 25(4A):2883–2919.

Benaouali, A. and Kachel, S. (2019). Multidisciplinary design optimization of aircraft wing using commercial software integration. *Aerospace Science and Technology*, 92:766–776.

Bull, A. D. (2011). Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(10).

Charisopoulos, V., Chen, Y., Davis, D., Díaz, M., Ding, L., and Drusvyatskiy, D. (2019). Low-rank matrix recovery with composite optimization: good conditioning and rapid convergence. *arXiv preprint arXiv:1904.10020*.

Cramer, E. J., Dennis, Jr, J. E., Frank, P. D., Lewis, R. M., and Shubin, G. R. (1994). Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, 4(4):754–776.

Damianou, A. and Lawrence, N. (2013). Deep Gaussian processes. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, pages 207–215.

Drusvyatskiy, D. and Paquette, C. (2019). Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 178(1-2):503–558.

Frazier, P. I. (2018). A Tutorial on Bayesian Optimization. *arXiv preprint arXiv:1807.02811*.

Fu, M. C. and Henderson, S. G. (2017). History of seeking better solutions, aka simulation optimization. In *2017 Winter Simulation Conference (WSC)*, pages 131–157. IEEE.

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

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

Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial. *arXiv preprint arXiv:1604.00772*.

Hattrick-Simpers, J. R., Gregoire, J. M., and Kusne, A. G. (2016). Perspective: composition–structure–property mapping in high-throughput experiments: turning data into knowledge. *APL Materials*, 4(5):053211.

Jamil, M. and Yang, X.-S. (2013). A literature survey of benchmark functions for global optimization problems. *arXiv preprint arXiv:1308.4008*.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

Kandasamy, K., Dasarathy, G., Schneider, J., and Póczos, B. (2017). Multi-fidelity Bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1799–1808, International Convention Centre, Sydney, Australia. PMLR.

Kim, S., Pasupathy, R., and Henderson, S. G. (2015). A guide to sample average approximation. In *Handbook of simulation optimization*, pages 207–243. Springer.

Li, H., Zhao, G., Johari, R., and Weintraub, G. Y. (2021). Interference, bias, and variance in two-sided marketplace experimentation: Guidance for platforms. *arXiv preprint arXiv:2104.12222*.

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Lizotte, D. J., Wang, T., Bowling, M. H., and Schuurmans, D. (2007). Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949.

Marque-Pucheu, S., Perrin, G., and Garnier, J. (2019). Efficient sequential experimental design for surrogate modeling of nested codes. *ESAIM: Probability and Statistics*, 23:245–270.

Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 27 LNCS, pages 400–404. Springer.

Osorio, C. and Bierlaire, M. (2013). A simulation-based optimization framework for urban transportation problems. *Operations Research*, 61(6):1333–1345.

Poloczek, M., Wang, J., and Frazier, P. (2017). Multi-information source optimization. In *Advances in Neural Information Processing Systems*, pages 4288–4298.

Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Ryzhov, I. O. (2016). On the convergence rates of expected improvement methods. *Operations Research*, 64(6):1515–1528.

Sadowski, M. and Jones, D. (2009). The sequence–structure relationship and protein function prediction. *Current opinion in structural biology*, 19(3):357–362.

Sandberg, I. (1978). On the mathematical foundations of compartmental analysis in biology, medicine, and ecology. *IEEE transactions on Circuits and Systems*, 25(5):273–279.

Sekhon, B. S. and Saluja, V. (2011). Biosimilars: an overview. *Biosimilars*, 1(1):1–11.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Shapiro, A. (2003). On a class of nonsmooth composite functions. *Mathematics of Operations Research*, 28(4):677–692.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 4, pages 2951–2959.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Swersky, K., Snoek, J., and Adams, R. P. (2013). Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 2004–2012.

Toscano-Palmerin, S. and Frazier, P. I. (2018). Bayesian optimization with expensive integrands. *arXiv preprint arXiv:1803.08661*.

Uhrenholt, A. K. and Jensen, B. S. (2019). Efficient Bayesian optimization for target vector estimation. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2661–2670.

Vazquez, E. and Bect, J. (2010). Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and inference*, 140(11):3088–3095.

Wang, X., Gong, X., Geng, N., Jiang, Z., and Zhou, L. (2020). Metamodel-based simulation optimisation for bed allocation. *International Journal of Production Research*, 58(20):6315–6335.

Wang, Z. and Jegelka, S. (2017). Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR.

Williams, B. J., Santner, T. J., and Notz, W. I. (2000). Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, pages 1133–1152.

Wilson, J., Hutter, F., and Deisenroth, M. (2018). Maximizing acquisition functions for bayesian optimization. *Advances in Neural Information Processing Systems*, 31:9884–9895.

Wu, J. and Frazier, P. (2016). The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 3126–3134.

Wu, J., Toscano-Palmerin, S., Frazier, P. I., and Wilson, A. G. (2019). Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*.

Zhilinskas, A. (1975). Single-step Bayesian search method for an extremum of functions of a single variable. *Cybernetics*, 11(1):160–166.