BINOCULARS for Efficient, Nonmyopic Sequential Experimental Design

Shali Jiang [*1] **Henry Chai** [*1] **Javier González** [2] **Roman Garnett** [1]

Abstract

Finite-horizon sequential experimental design (SED) arises naturally in many contexts, including hyperparameter tuning in machine learning among more traditional settings. Computing the optimal policy for such problems requires solving Bellman equations, which are generally intractable. Most existing work resorts to severely myopic approximations by limiting the decision horizon to only a single time-step, which can underweight exploration in favor of exploitation. We present BINOCULARS: **B**atch-**I**nformed **NO**nmyopic **C**hoices, **U**sing **L**ong-horizons for **A**daptive, **R**apid **S**ED, a general framework for deriving efficient, *nonmyopic* approximations to the optimal experimental policy. Our key idea is simple and surprisingly effective: we first compute a one-step optimal batch of experiments, then select a single point from this batch to evaluate. We realize BINOCULARS for Bayesian optimization and Bayesian quadrature – two notable SED problems with radically different objectives – and demonstrate that BINOCULARS significantly outperforms myopic alternatives in real-world scenarios.

1. Introduction

Many real-world problems can be framed as finite-horizon sequential experimental design (SED), wherein an agent adaptively designs a prespecified number of experiments seeking to maximize some data-dependent utility function. The optimal policy for SED can be formulated as dynamic programming (DP), which balances the inherent tradeoff between exploitation (immediately advancing the goal) and exploration (learning for the future). However, this optimal policy is intractable even for simple problems (Powell, 2010). Common approximation schemes include rollout, Monte Carlo tree search (Bertsekas, 2017; Powell, 2010), or simply artificially limiting the horizon, known as a *myopic* approximation.

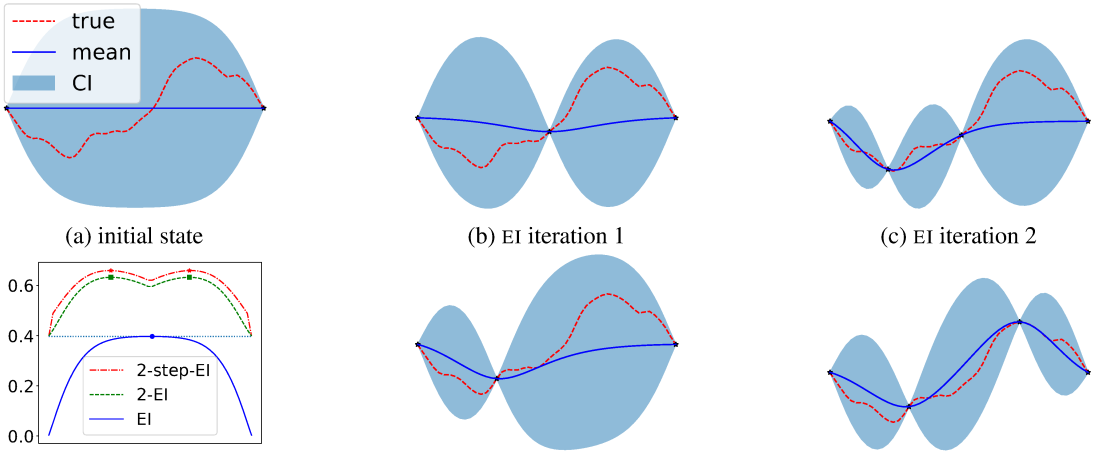
In this work, we propose a novel method for *efficient and nonmyopic* SED, called BINOCULARS: **B**atch-**I**nformed **NO**nmyopic **C**hoices, **U**sing **L**ong-horizons for **A**daptive, **R**apid **S**ED. BINOCULARS is inspired by the fact that the optimal batch (or non-adaptive) design is an approximation to the optimal sequential (or adaptive) design. In fact, the optimal adaptive and non-adaptive designs are exactly the same in some notable cases where the data utility does not depend on the observed outcomes, such as maximizing information gain for a fixed Gaussian process (GP) (Krause and Guestrin, 2007). Even when this is not the case, we show that the optimal batch expected utility is a lower bound of the optimal sequential expected utility. Furthermore, it is always as tight as the one-step optimal policy's implied expected utility. Motivated by this insight, BINOCULARS iteratively computes an optimal batch of designs, then chooses one point from this batch. While many existing methods construct batch policies by simulating a sequential policy (Ginsbourger et al., 2010; Desautels et al., 2014; Jiang et al., 2018), BINOCULARS goes the other way and "reduces" sequential design to batch design.

BINOCULARS is a general framework applicable to any SED problem. In this paper, we realize this framework on two important yet *fundamentally different* SED tasks: Bayesian optimization (BO) (Kushner, 1964; Močkus, 1974; Shahriari et al., 2016) and Bayesian quadrature (BQ) (Larkin, 1972; Diaconis, 1988; O'Hagan, 1991). In BO, an agent repeatedly queries an expensive function seeking its global optimum, whereas in BQ the goal is to estimate an intractable integral of the function.

For both problems, many popular policies are myopic: examples include expected improvement (EI) for BO (Močkus, 1974) and uncertainty sampling (UNCT) for BQ (Gunter et al., 2014). These are all one-step optimal for maximizing particular utility functions in expectation. While they are computationally efficient and give reasonable empirical results, they are liable to suffer from myopia and overexploitation. Nonmyopic alternatives have recently been applied to BO (González et al., 2016b; Lam et al., 2016; Yue

[*]Equal contribution [1]Department of Computer Science and Engineering, Washington University in Saint Louis, Saint Louis, Missouri, USA [2]Microsoft Research Cambridge, Cambridge, UK. Correspondence to: Shali Jiang <jiang.s@wustl.edu>, Henry Chai <hchai@wustl.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.



(a) initial state (b) EI iteration 1 (c) EI iteration 2

(d) EI, 2-EI and 2-step-EI (e) 2-EI iteration 1 (f) 2-EI iteration 2

Figure 1: An illustration of our proposed nonmyopic method applied to BO. (a) A function in $[-1, 1]$ drawn from a GP where the two end points are known to be zero. (b) and (c) show two iterations of BO with the EI acquisition function. (d) EI, 2-EI and 2-step-EI curves with their respective maximizers. (e) and (f) show two iterations of BO where the first point is chosen from the two points maximizing 2-EI, and the second one is chosen by maximizing EI (conditioned on the observation in iteration one).

and Al Kontar, 2019): while results are promising, these are typically costly to compute.

Our contributions can be summarized as follows: (1) We propose a general framework for efficient and nonmyopic SED with finite horizons, inspired by the close connection between optimal sequential and batch designs. (2) We realize the framework on two important SED problems: Bayesian optimization and Bayesian quadrature. This represents the first nonmyopic policy proposed for BQ. (3) We conduct thorough experiments demonstrating that the proposed method significantly outperforms the myopic baselines and is competitive with (if not better than) state-of-the-art nonmyopic alternatives, while being much more efficient.

2. BINOCULARS

We first illustrate the intuition behind BINOCULARS and provide explicit mathematical justification. We then realize BINOCULARS for two specific SED scenarios: BO and BQ. Throughout this work, we make extensive use of Gaussian processes (GPs): a GP defines a probability distribution over functions, where the joint distribution of the function's values at finitely many locations is multivariate normal; for more details, see (Rasmussen and Williams, 2006).

Intuition. Consider the BO example in Fig. 1, where we wish to maximize a one-dimensional objective function over an interval, conditioned on initial observations at the boundary. Suppose we have two more function evaluations left. The myopic EI policy would greedily pick the middle point first, followed by a point bisecting the left half of the domain. The resulting choices completely ignore the right half of the domain, which is where the maximum happens to lie.

Now consider the following alternative for designing the observations: we first construct the optimal *batch* of size two (2-EI). These points can be determined relatively efficiently as recursion is not required and reflect a better approximation of the remainder of the optimization than just looking one step ahead. We then pick a point from this batch (how the point is selected will be addressed later) and use EI to choose the final point given the result. This policy results in well-distributed queries and better performance. We can compare these decisions with the *optimal* (but expensive) policy maximizing the full lookahead expected utility (2-step-EI in Fig. 1(d)): our choices are nearly perfect.

2.1. The Optimal Adaptive Policy

Consider a general SED problem with a finite horizon, T. Let the design space be $\mathcal{X}$, response space be $\mathcal{Y}$; for $x \in \mathcal{X}, y \in \mathcal{Y}$ and $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$, let $p(y \mid x, \mathcal{D})$ be a probabilistic model; and let $u(\mathcal{D})$ be some utility function of observed data $\mathcal{D}$. Define $u(y \mid x, \mathcal{D}) = u\big(\mathcal{D} \cup (x, y)\big) - u(\mathcal{D})$ as the marginal gain in utility after observing y from experiment x when $\mathcal{D}$ has already been observed. Let $Q_k(x \mid \mathcal{D})$ be the expected utility of designing experiment x after observing $\mathcal{D}$ when there are k steps remaining, assuming all later decisions are optimal. $Q_k(x \mid \mathcal{D})$ can be expressed in the form of a Bellman equation as follows:

$$Q_k(x \mid \mathcal{D}) = \mathbb{E}_y[u(y \mid x, \mathcal{D})] + \mathbb{E}_y\Big[\max_{x'} Q_{k-1}\big(x' \mid \mathcal{D} \cup \{(x, y)\}\big)\Big], \quad (1)$$

where the expectation is taken with respect to $p(y \mid x, \mathcal{D})$. The optimal (expected-case) policy is to observe

$$x^* = \operatorname*{argmax}_x Q_{T-i}(x \mid \mathcal{D}_i), \quad (2)$$

where $\mathcal{D}_i$ is the observed set at iteration i. The optimal policy is intractable for any moderately large horizon; in general, the complexity is $\mathcal{O}\left(|\mathcal{X}|^T|\mathcal{S}|^T\right)$, where $\mathcal{S} = \{\mathcal{D} \mid \mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}\}$, and in many settings $\mathcal{X}$ and/or $\mathcal{S}$ are uncountable. Thus, we must find some tractable approximation to proceed. A common solution is to limit the horizon to some manageable value ℓ, e.g. $\ell = 1$ or 2. This is called *ℓ-step lookahead,* and is computationally efficient but *myopic* as it severely limits our view of the future: it does not plan ahead and can thus make suboptimal tradeoffs between exploration and exploitation.

2.2. Nonmyopic Approximation via the Optimal Non-Adaptive Policy

Suppose T experiments $X = \{x_1, \ldots, x_T\}$ must be designed *simultaneously* given current observations $\mathcal{D}$. The expected marginal utility of the resulting observations is

$$Q(X \mid \mathcal{D}) = \mathbb{E}_Y[u(Y \mid X, \mathcal{D})], \tag{3}$$

where the expectation is taken over the joint distribution of $Y = \{y_1, \ldots, y_T\}$, $p(Y \mid X, \mathcal{D})$. Rewriting (3) by decomposing X into x_j and X_{-j} where $X_{-j} = X \setminus \{x_j\}$, we have (by telescoping sum)

$$Q(X \mid \mathcal{D}) = \mathbb{E}_{y_j}[u(y_j \mid x_j, \mathcal{D})] + \mathbb{E}_{y_j}\Big[Q\big(X_{-j} \mid \mathcal{D} \cup \{(x_j, y_j)\}\big)\Big]. \tag{4}$$

The derivation of this decomposition can be found in the supplement. Let $X^* \in \arg\max_X Q(X \mid \mathcal{D})$ be an optimal batch of experiments. For any $x_j^* \in X^*$,

$$\mathbb{E}_{y_j^*}\Big[Q\big(X_{-j}^* \mid \mathcal{D} \cup \{(x_j^*, y_j^*)\}\big)\Big] = \max_{X_{-j}} \mathbb{E}_{y_j^*}\Big[Q\big(X_{-j} \mid \mathcal{D} \cup \{(x_j^*, y_j^*)\}\big)\Big], \tag{5}$$

as otherwise we could construct a batch with higher utility than $Q(X^* \mid \mathcal{D})$. Therefore, given that the expected reward of the entire batch can be decomposed using (4), choosing any experiment $x^* \in X^*$ is equivalent to solving the following optimization: $x^* \in \arg\max_x B(x \mid \mathcal{D})$ where

$$B(x \mid \mathcal{D}) = \mathbb{E}_y[u(y \mid x, \mathcal{D})] + \max_{X' : |X'| = T-1} \mathbb{E}_y\Big[Q\big(X' \mid \mathcal{D} \cup \{(x, y)\}\big)\Big]. \tag{6}$$

Comparing (6) and the Bellman equation (1), we see two differences: 1) the expectation and maximization are exchanged in the future utility term and 2) the adaptive utility is replaced by a non-adaptive counterpart. As such, (6) is clearly a *lower bound of the true expected utility*:

$$\begin{aligned}\max_{X' : |X'| = T-1} \mathbb{E}_y\Big[Q\big(X' \mid \mathcal{D} \cup \{(x, y)\}\big)\Big] &\leq \mathbb{E}_y\left[\max_{X' : |X'| = T-1} Q\big(X' \mid \mathcal{D} \cup \{(x, y)\}\big)\right] \\ &\leq \mathbb{E}_y\left[\max_{x'} Q_{T-1}\big(x' \mid \mathcal{D} \cup \{(x, y)\}\big)\right]. \end{aligned} \tag{7}$$

This is illustrated in Fig. 1(d): 2-step-EI corresponds to (1), and 2-EI to (6). Note that while the one-step optimal (myopic) policy also optimizes a lower bound of the true expected utility, the lower bound in (6) is always at least as tight as the lower bound optimized by the myopic policy. An interesting open question is the tightness of this bound, closely related to the so-called *adaptivity gap* (Jiang et al., 2018; Krause and Guestrin, 2007).

The similarity between these formulations provides mathematical justification for using (6) to approximate the optimal policy. Note that (6) is exactly equal to (1) if the remaining experiments ever become conditionally independent given the observed data, in which case there is no advantage to adaptation.

Algorithm 1 BINOCULARS

Input: design space $\mathcal{X}$, response space $\mathcal{Y}$, model $p(y \mid x, \mathcal{D})$, utility function $u(y \mid x, \mathcal{D})$, budget T
Output: $\mathcal{D}$, a sequence of experiments and observations
for $i \leftarrow 0$ to $T - 1$ **do**
  Compute the optimal batch X^* of size $T - i$
  Pick an experiment $x^* \in X^*$ and observe response y^*
  Augment $\mathcal{D} = \mathcal{D} \cup \{(x^*, y^*)\}$

BINOCULARS is summarized in Algorithm 1. The primary computational cost comes from computing the optimal batch, a high-dimensional optimization problem. For the examples considered below (BO and BQ), this optimization can be done using gradient-based methods and we show empirically that BINOCULARS runs much faster than previously proposed nonmyopic methods (see section 6). Note that while we do use a batch method, it is only as a subroutine. Algorithm 1 is for *sequential* experimental design: in each iteration, we only observe the outcome of one experiment.

3. BINOCULARS for Bayesian Optimization

Consider the task: $x^* = \arg\max_{x \in \mathcal{X}} f(x)$; in this paper, we model f with a GP. Suppose we have a budget of T function evaluations. Once the budget has been expended, we recommend the point with the highest observed value as the maximizer of f. In this setting, our goal is to *sequentially* select a set $X = \{x_1, x_2, \ldots, x_T\}$ of T points from $\mathcal{X}$ such that $\max\{y_j\}$ is maximized, where $y_j = f(x_j)$.

Let $\mathcal{D}_0$ be a set of initial observations, and $y_0 = \max_{(x,y)\in\mathcal{D}_0} y$ is the initial best observed value. We define the utility function as the improvement over y_0:

$$u(Y \mid X, \mathcal{D}_0) = \left(\max_{x_j \in X} y_j - y_0\right)^+, \tag{8}$$

where $c^+ = \max(c, 0)$. Defining the utility as improvement allows us to write the expected utility as a Bellman equation with the same form as (1):

$$EI_k(x) = EI_1(x) + \mathbb{E}_y\left[\max_{x'} EI_{k-1}(x' \mid x, y)\right], \tag{9}$$

where $EI_k(x)$ is the expected improvement of k adaptive decisions starting from x, and $EI_{k-1}(x' \mid x, y)$ is an expectation taken over the posterior belief of f after further conditioning on the observation (x, y) and replacing y_0 by $\max(y_0, y)$. The derivation of (9) can be found in the supplement. Observe that $\arg\max_x EI_1(x)$ exactly corresponds to the popular *expected improvement* (EI) policy (Močkus, 1974), which is one-step optimal; $EI_2(x)$ is already analytically intractable as it requires an expensive numerical integration: the integrand is $\max_{x'} EI_1(x' \mid x, y)$ and entails global optimization!

To apply BINOCULARS, we optimize the batch EI objective, also known as q-EI, via the recently developed reparameterization trick and Monte Carlo approximation (Wang et al., 2016). Then we pick a point from the optimal batch; how to pick this point is discussed later. BINOCULARS trivially extends to other utility functions such as knowledge gradient (Wu and Frazier, 2016), probability of improvement (Kushner, 1964) and predictive entropy (Shah and Ghahramani, 2015) by replacing q-EI appropriately.

4. BINOCULARS for Bayesian Quadrature

Consider a non-analytic integral of the form $Z = \int f(x)\pi(x)\,\mathrm{d}x$, where $f(x)$ is a likelihood function and $\pi(x)$ is a prior. Such integrals frequently occur in Bayesian inference (e.g., Bayesian model selection and averaging). Bayesian quadrature operates by placing a GP on the integrand and then minimizing the posterior variance of Z:

$$\mathrm{Var}[Z \mid X] = \iint K_X(x, x')\pi(x)\pi(x')\,\mathrm{d}x\,\mathrm{d}x', \tag{10}$$

where $X = \{x_1, x_2, \ldots, x_T\}$ is a set of T points that needs to be optimized, and $K_X(x, x')$ is the posterior covariance after conditioning on observations at X. If the GP hyperparameters are fixed, the optimal design $X^* = \operatorname{argmin}_X \mathrm{Var}[Z \mid X]$ can be precomputed, as the posterior covariance of a GP does not depend on the observed values $f(X)$; this effectively eliminates the need for sequential experimental design in this setting.

However, in general the hyperparameters are not fixed *a priori*, but are instead learned iteratively in light of new observations. Furthermore, when the integrand is known to be *positive* (e.g., a likelihood function), it is often a good practice to place a GP on some non-linear transformation of f, such as $\sqrt{f}$ or $\log(f)$ (Osborne et al., 2012; Gunter et al., 2014; Chai and Garnett, 2019). As a result, the posterior GP must be approximated (e.g., by moment matching), which causes the posterior covariance to depend on the observed values. In these cases adaptive sampling becomes critical.

The adaptive version of $\mathrm{Var}[Z \mid X]$ is computationally expensive to evaluate so Gunter et al. (2014) proposed the use of *uncertainty sampling* (UNCT) (Lewis and Gale, 1994; Settles, 2010) as a surrogate, i.e., sequentially evaluating the location with the largest variance. This greedily minimizes the entropy of the integrand instead of the integral.

Formally, we use the differential entropy of the conditional distribution $p(Y \mid X)$ as the utility function:

$$H\big(p(Y \mid X)\big) = \tfrac{1}{2}\log\Big(\det\big(2\pi e\, K(X, X)\big)\Big). \tag{11}$$

Using the chain rule for differential entropy, this quantity can be expressed in the same form as (1):

$$H\big(p(Y \mid X)\big) = H\big(p(y_j \mid x_j)\big) + \mathbb{E}_{y_j}\Big[H\big(p(Y_{-j} | X_{-j}, x_j, y_j)\big)\Big]. \tag{12}$$

Note that $\arg\max_{x_j} H(p(y_j \mid x_j))$ corresponds to the sequential uncertainty sampling policy. To apply BINOCULARS for BQ, we must find $\arg\max_X H(p(Y \mid X))$, which is the mode of a determinantal point process (DPP) (Kulesza and Taskar, 2012) defined over $q = |X|$ points. [1] This can be done using gradient-based optimization. Note that this formulation can be applied to active learning of GPs, where uncertainty sampling is also a common strategy.

Practical Considerations. Some practical issues arise when applying BINOCULARS to real problems. First, given an optimal batch, how should one select a point from this batch? We considered several options: selecting the point with the highest expected immediate reward or randomly selecting a point, either uniformly or proportional to its expected immediate reward. Empirically, we found that "best" and "proportional sampling" perform similarly while "uniform sampling" performs the worst.

Second, given that BINOCULARS is only an approximation to the optimal policy, it is not necessarily true that setting q to the exact remaining budget is the best. In theory, if the model is perfect, then full lookahead is optimal. However, in practice, the model is always wrong and thus planning too far ahead could actually harm the empirical performance (Yue and Al Kontar, 2019). Furthermore, smaller values of

[1] This connection is purely theoretical: our method uses no properties of DPPs but it is the basis of our BQ naming convention.

q result in more efficient computation. We empirically study the choice of q in section 6.

5. Related Work

General introductions to approximate dynamic programming (DP) can be found in Bertsekas (2017); Powell (2010). On the subject of nonmyopic BO, Osborne et al. (2009) derived the optimal policy for BO, and demonstrate that it is possible to approximately compute the two-step lookahead policy for low-dimensional functions and that it generally performs better than the one-step policy. Ginsbourger and Le Riche (2010) also derived the optimal policy and gave an explicit example where two-step EI is better than one-step EI in expectation with a desired degree of statistical significance. González et al. (2016b) proposed a nonmyopic approximation of the optimal policy, known as GLASSES, by simulating future decisions using a batch BO method.[2] Jiang et al. (2017; 2018) proposed a nonmyopic policy for (batch) active search, which can be understood as a special case of BO with cumulative reward, using a similar idea. Lam et al. (2016) proposed to use rollout for BO, a classic approximate DP method (Bertsekas, 2017). Yue and Al Kontar (2019) presented theoretical justification for rollout, and gave theoretical and practical guidance on how to choose the rollout horizon. Ling et al. (2016) proposed a branch-and-bound near-optimal policy for GP planning assuming that the reward function is Lipschitz continuous, and applied it to BO and active learning. Wu and Frazier (2019) proposed a gradient-based optimization of two-step EI, but each evaluation of two-step EI still requires a quadrature subroutine with an expensive integrand: optimization of one-step EI.

Of these, GLASSES and rollout are most related to BINOCULARS. GLASSES's acquisition function shares almost the same form as (6), except the future batch X' is constructed using a heuristic batch policy, instead of optimized with the q-EI objective. The batch policy adds points one by one by optimizing the sequential EI function penalized at locations already added to the batch (González et al., 2016a), and the expected utility of the chosen batch is estimated using expectation propagation.

Rolling out two steps using EI as the heuristic policy is exactly equivalent to the two-step lookahead policy, up to quadrature error. Mathematically, the rollout acquisition function can also be written in a similar form as (6), except X' is adaptively constructed, depending on sampled values of y instead of globally (irrespective of y) constructed or optimized as in GLASSES and BINOCULARS. Both rollout and GLASSES are very expensive to compute.

While we are unaware of any existing work on nonmyopic BQ, there has been some prior work on nonmyopic active learning of GPs. Krause and Guestrin (2007) derived the adaptivity gap for active learning of GPs under two utility functions. They also proposed a nonmyopic method for active learning of GPs which separates the process into an exploration phase and an exploitation phase. They considered different acquisition functions for the exploration phase; notably, the implicit exploration (IE) method is comparable to the uncertainty sampling baseline in subsection 6.2. Hoang et al. (2014) developed a method for active learning of GPs that does away with separate exploration and exploitation phases and instead naturally trades off between the two. Their proposed policy, ε-BAL, approximates the solution to the Bellman formulation of the active GP learning problem using a truncated sampling method. They analyzed the theoretical performance of their method and developed a pruning-based anytime version of their method.

The setting of our BQ work (integration of non-negative integrands) and active learning of GPs appear related yet are fundamentally different. The cited works focus exclusively on learning the hyperparameters of the GP. In our setting, the use of a transformation to model non-negativity introduces adaptivity beyond the GP hyperparameters: even if the true GP hyperparameters are known *a priori*, the nonlinear transformation causes the approximate GP posterior to depend on the observed values.

6. Experiments

We designed our experiments to broadly test the performance and computational cost of BINOCULARS relative to notable myopic and nonmyopic baselines for BO and BQ. We also conducted a thorough exploration of the BINOCULARS design choices: the number of steps to look ahead and how to select a point from the optimal batch.

The primary takeaways of our experimental results are that BINOCULARS outperforms myopic baselines while running only slightly slower and is at least as good as previously proposed nonmyopic methods while running orders of magnitude faster. This places it *on the Pareto front* of the running time–performance tradeoff in policy design. Furthermore, BINOCULARS clearly demonstrates *distinctively nonmyopic behavior* on both BO and BQ tasks, two entirely different SED problems.

We use the following nomenclature to describe BINOCULARS: our nonmyopic BO method will be denoted as "q.EI.s" or "q.EI.b", where q is the batch size and "s" represents sampling from the batch while "b" means choosing the "best." For BQ, we replace "EI" with "DPP." In addition to the myopic methods, EI and UNCT, we also compare against rollout for both tasks and GLASSES for BO.[3] Each rollout method

[2] The name BINOCULARS is inspired by GLASSES.

[3] We did not compare against a BQ-equivalent of GLASSES as no such method has been published.

Table 1: Average GAP over 100 repeats on "hard" synthetic functions.

	Rand	EI	2.EI.b	2.EI.s	3.EI.b	3.EI.s	4.EI.b	4.EI.s	10.EI.b	10.EI.s	12.EI.s	15.EI.s
eggholder	0.498	0.613	0.614	0.633	0.604	0.657	0.646	*0.694*	0.622	*0.704*	**0.738**	*0.694*
dropwave	0.486	0.439	0.507	0.531	0.473	*0.552*	0.467	0.514	0.397	*0.591*	**0.598**	*0.585*
shubert	0.355	0.408	0.366	*0.441*	*0.394*	**0.507**	0.388	*0.484*	0.305	*0.455*	*0.479*	*0.465*
rastrigin4	0.374	0.801	0.769	0.775	0.817	*0.821*	**0.840**	0.805	0.797	0.804	0.793	0.799
ackley2	0.358	0.821	0.825	0.823	0.819	*0.869*	0.812	*0.872*	0.801	**0.892**	*0.886*	*0.888*
ackley5	0.145	0.509	0.544	0.509	0.601	0.550	0.596	0.592	**0.636**	0.606	*0.627*	*0.626*
bukin	0.600	*0.849*	0.856	0.855	*0.872*	*0.859*	0.864	*0.865*	**0.878**	0.850	0.829	*0.853*
shekel5	0.038	0.286	0.311	0.320	0.330	0.343	0.342	0.344	*0.374*	*0.373*	*0.358*	**0.395**
shekel7	0.045	0.268	0.346	0.313	0.349	0.325	0.352	0.370	*0.399*	0.358	**0.412**	*0.386*
Average	0.322	0.555	0.571	0.578	0.584	0.609	0.590	0.616	0.579	*0.626*	**0.635**	*0.632*

is denoted as "q.R.n", where q represents the number of steps to roll out, and n is the number of samples used to estimate the expectations encountered in each step. Each GLASSES method is denoted as "q.G" where q represents the size of the simulated batch. We use DIRECT (Jones, 2009) to optimize the GLASSES and rollout acquisition functions, following González et al. (2016b). For all nonmyopic methods, when the remaining budget $r < q$, we set $q = r$. Thus the final decision is always made (optimally) with one-step lookahead.

For all experiments, we start with $2d$ randomly-sampled observations and perform $20d$ further iterations, where d is the function's dimensionality. Unless otherwise noted, all results presented are aggregated over 100 repeats with different random initializations. For all tabulated results, the best method is indicated in bold and the entries not significantly worse than the best (under a one-sided paired Wilcoxon signed-rank test with $\alpha = 0.05$) are in blue italics.

6.1. BO Results

We implemented our nonmyopic BO policy and all baselines using `BoTorch`,[4] which contains efficient EI and q-EI implementations. We present experiments for two rollout variants: "2.R.10" and "3.R.3." As we will see, rolling out with horizon two is already very expensive even for just ten y samples. Gauss–Hermite quadrature is used for rollout as in Lam et al. (2016). We also present experiments for two GLASSES variants: "2.G" and "3.G".[5]

We use GPs with a constant mean and a Matérn $5/2$ ARD kernel to model the objective function, the default in `BoTorch`. We tune hyperparameters every iteration by maximizing the marginal likelihood using L-BFGS-B. We also maximize the q-EI acquisition function with L-BFGS-B. Complete details can be found in our attached code. We use the GAP measure to evaluate the performance: $\text{GAP} = (y_i - y_0) \,/\, (y^* - y_0)$, where y_i's are maximum observed values and y^* is the true optimal value; we convert all problems to maximization problems by negating if necessary.

Synthetic Functions. In this section, we focus on demonstrating the superior performance of our method over EI on nine "hard" benchmark functions. These nine functions are selected by first running experiments on 31 functions[6] with 30 repeats (see Table 1 in the supplement). We then select the ones where EI terminates with average GAP < 0.9. We believe nonmyopic methods are more advantageous on challenging functions; by first identifying these hard problems, we will gain more insight into the various policies. To put the BO performance into perspective, we also include a comparison against a random baseline, "Rand."

Table 1 shows the average GAP at termination. For the results in Table 1, the p-values of the best BINOCULARS variant for each function against EI in *descending* order are 0.065, 0.025, 0.0030, 3.0e-5, ... Thus, after applying the Bonferroni correction to account for the 10 variants of BINOCULARS ($p \leq \alpha \,/\, 10$), the best variant of our method remains significantly better than EI for 7 out of the 9 "hard" functions. After applying the Holm-Bonferroni correction to our aggregated results, every variant of BINOCULARS remains significantly better than EI at the $\alpha = 0.05$ level and all but 2.EI.b at the $\alpha = 0.01$ level.

We summarize the results as follows: (1) All q.EI.s variants perform significantly better than EI on average, with 12.EI.s being the best and outperforming EI by a large margin. (2) The q.EI.s variants are consistently better than the q.EI.b variants (for better spacing we did not show results for 12.EI.b and 15.EI.b). (3) The performance of our method generally improves as we increase q, up to 12.

Perhaps more interestingly, we can clearly observe the nonmyopic behavior of 12.EI.s as shown in Figure 2: it is ini-

[4] https://github.com/pytorch/botorch

[5] With help from the authors of (González et al., 2016b), we implemented an advanced version of GLASSES in `BoTorch`, using quasi Monte Carlo instead of expectation propagation to estimate the expected improvement of the batch, a standard practice for computing qEI in state of the art BO packages such as `BoTorch`.

[6] https://www.sfu.ca/~ssurjano/optimization.html

Table 2: Average GAP over 50 repeats on real functions.

	EI	2.EI.s	3.EI.s	4.EI.s	6.EI.s	8.EI.s	2.G	3.G	2.R.10	3.R.3
SVM	0.738	*0.913*	**0.940**	*0.911*	*0.937*	0.834	*0.881*	0.898	*0.930*	*0.928*
LDA	0.956	**1.000**	*0.996*	*0.993*	0.982	*0.995*	*1.000*	*0.999*	*0.999*	*1.000*
LogReg	0.963	*0.998*	*1.000*	*0.999*	*0.999*	**1.000**	0.989	0.911	0.965	0.948
NN Boston	*0.470*	*0.467*	*0.478*	*0.460*	*0.502*	*0.467*	0.455	**0.512**	*0.503*	*0.482*
NN Cancer	0.665	0.627	0.654	0.686	0.700	0.686	**0.806**	*0.755*	0.708	0.698
Robot pushing 3d	0.928	*0.960*	*0.962*	*0.957*	**0.962**	*0.961*	*0.955*	0.951	*0.955*	*0.954*
Robot pushing 4d	*0.730*	0.726	0.695	0.695	0.736	0.697	*0.765*	**0.786**	*0.770*	*0.745*
Average	0.779	*0.813*	*0.818*	0.815	*0.831*	0.806	**0.836**	*0.830*	*0.833*	*0.822*

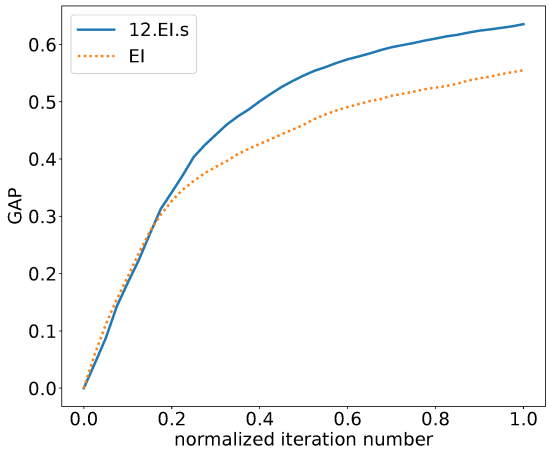


Figure 2: Average GAP over nine synthetic functions demonstrating the nonmyopic behavior of 12.EI.s.

tially outperformed by the myopic EI as it explores the space. However, our method catches up to EI at ∼20% of the budget (on average) as it transitions to exploiting its findings until finally, it outperforms EI by a large margin. This behavior indicates that our method seamlessly navigates the exploration/exploitation tradeoff without the need for any external intervention.

Real World Functions. In this section, we compare our method against popular nonmyopic baselines: rollout and GLASSES. We present results on hyperparameter tuning functions used by Snoek et al. (2012); Wang and Jegelka (2017); Malkomes and Garnett (2018). These functions are evaluated on a predefined grid, so we first compute all policies (except EI) using continuous optimization, then pick the closest point from the grid.

Table 2 shows the results averaged over 50 repeats. We only show the "sampling" variants of our method; full results can be found in Table 3 in the appendix. First we see again all q.EI.s variants outperform EI by a large margin, with $q = 6$ achieving the best results. Comparing 6.EI.s with the nonmyopic baselines, 2.G is the best, but the difference of 0.005 is negligible; the p-value under a one-sided paired signed-rank test for 6.EI.s against 2.G is 0.4257.

We now focus on comparing the time cost of the tested methods. Figure 3 shows the average GAP versus average time per iteration; the average is taken over 350 experiments (seven functions with 50 repeats each); error bars are also plotted. We again see that our methods are not significantly different from rollout and GLASSES in terms of GAP performance, but are considerably faster in terms of average time cost per iteration (note the log scale on the time axis). Clearly, our method lies on the Pareto front in terms of computational cost and performance.

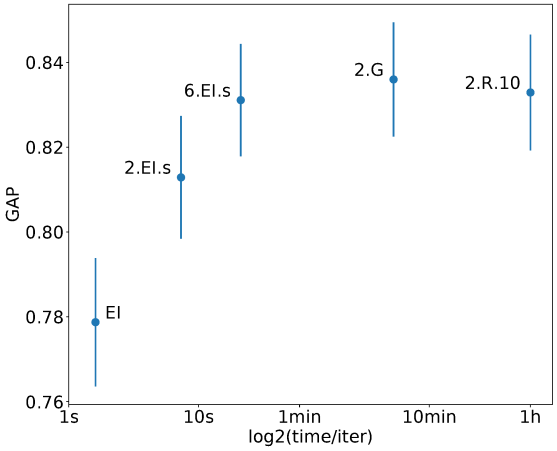
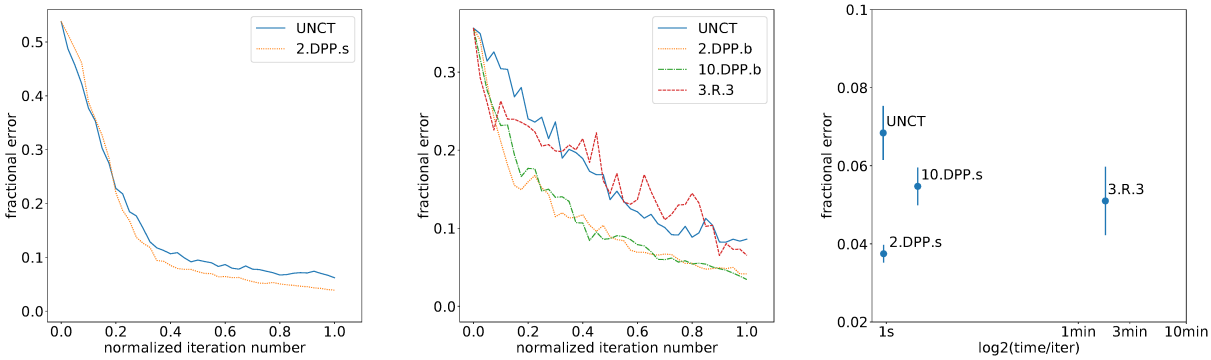


Figure 3: mean GAP with error bars at termination versus time per iteration (in log scale) averaged over the seven real functions.

We also attempted to compare with the recently published practical two-step EI method (Wu and Frazier, 2019), which is intended to be a more efficient version of our 2.R.n; the difference is first- versus zeroth-order optimization of the acquisition function. Our implementation of rollout supports gradient-based optimization thanks to automatic differentiation. However, we did not find it considerably faster than using DIRECT. We leave it to future work to optimize the implementation and compare with our method.

Table 3: Median fractional error values over 100 repeats on all BQ functions.

	UNCT	2.DPP.b	3.DPP.b	10.DPP.b	2.DPP.s	3.DPP.s	10.DPP.s	2.R.10	3.R.3
cont	0.045	0.052	0.055	0.059	0.039	0.037	**0.029**	0.036	0.045
corner	0.265	0.206	0.137	0.065	**0.047**	0.078	0.132	0.074	0.063
discont	*0.523*	*0.511*	*0.488*	**0.446**	0.572	0.610	0.590	0.537	0.577
Gauss	*0.004*	0.004	0.005	0.006	**0.003**	*0.003*	*0.003*	0.004	*0.003*
MM	0.254	0.207	0.203	0.207	0.221	0.161	0.177	*0.110*	**0.086**
prod	0.007	0.007	*0.007*	0.007	0.007	**0.006**	*0.006*	0.012	0.012
GP	0.231	0.082	**0.057**	0.077	*0.069*	*0.073*	0.116	0.283	0.248
DLA	0.019	*0.013*	0.025	*0.013*	*0.016*	*0.016*	0.033	*0.019*	**0.011**
Average	0.068	0.056	0.055	*0.041*	**0.037**	*0.043*	0.055	0.049	0.051



(a) (b) (c)

Figure 4: Median fractional error over 100 repeats against iterations or time per iteration (in log scale). (a) synthetic functions, (b) real functions, (c) all functions.

It is also possible to further improve rollout's performance using an adaptive rolling horizon in light of a recent study (Yue and Al Kontar, 2019), but it would be even more expensive to compute. In fact, Figure 1 in (Yue and Al Kontar, 2019) shows that with their adaptive horizon approach, the most frequently chosen horizon was two.

6.2. BQ Results

We implemented our nonmyopic BQ policy and all baselines using the GPML MATLAB package.[7] For all BQ experiments, we use the framework of Chai and Garnett (2019): we place GP priors on the log of the integrands as they are all nonnegative. We use GPs with a constant mean and a Matérn $^3/_2$ ARD kernel to model the integrands. We tune the GP hyperparameters after each observation by maximizing the marginal likelihood using L-BFGS-B. We also use L-BFGS-B to maximize the DPP likelihood. Complete details of our implementation can be found in our attached code.

We perform experiments on five standard benchmark synthetic functions[8] as well as one additional synthetic benchmark and two real model likelihood functions used by Chai et al. (2019). The additional synthetic benchmark is: $f(x) = \prod_{i=1}^{d} \frac{\sin(x_i)+\cos(3x_i))^2/2}{x_i^2/4+0.3}$; this function was included because of its multi-modal (MM) nature. We evaluate the performance of all methods using their fractional error: $|Z - \hat{Z}|/Z$ where $\hat{Z}$ is the estimate of the integral.

Figure 4(a) indicates that 2.DPP.s exhibits the same nonmyopic behavior as 12.EI.s: it initially lags behind but eventually overtakes the myopic UNCT, again suggesting a superior and automatic tradeoff of exploration and exploitation.

Table 3 shows the median fractional error at termination for all BQ experiments. Again, we analyzed the results in Table 3 to account for multiple comparisons: the p-values of the best BINOCULARS variant for each function against UNCT in *descending* order are 0.22, 0.16, 0.009, 0.0001, ... Thus BINOCULARS significantly outperforms UNCT on 5 out of 8 functions after applying the Bonferroni correction. After applying the Holm-Bonferroni correction to our aggregated results, every variant of BINOCULARS remains significantly

[7] http://gaussianprocess.org/gpml/code/matlab

[8] https://www.sfu.ca/~ssurjano/integration.html

better than UNCT at the $\alpha = 0.05$ level and all but 10.DPP.s at the $\alpha = 0.01$ level.

Fig. 4 shows the convergence of the fractional error as a function of both iterations and time per iteration (in log scale). These results corroborate many of the findings from our BO experiments: (1) All nonmyopic methods outperform UNCT on average with 2.DPP.s running only imperceptibly slower than UNCT. (2) Our proposed nonmyopic methods are competitive with, if not better than, rollout while running orders of magnitude faster.

We also note that in general, q.DPP.s variants tend to outperform q.DPP.b variants and increasing the batch size q does not consistently improve the performance.

The primary conclusion here is the same as for BO: BINOCULARS significantly and consistently outperforms myopic policies while only slightly increasing computational cost.

7. Conclusion and Future Work

We proposed BINOCULARS: an efficient, nonmyopic approximation framework for finite-horizon sequential experimental design. BINOCULARS computes an optimal batch, then picks a point from the batch. We gave an intuitive understanding and a mathematical justification for why this is a reasonable approximation. We applied BINOCULARS to Bayesian optimization and Bayesian quadrature, two entirely different problems, and empirically demonstrated that it significantly outperforms commonly used myopic policies while being much more efficient than popular nonmyopic alternatives.

As suggested by Yue and Al Kontar (2019), it would be useful to derive theories to guide the choice of lookahead horizon q for our method. Another interesting theoretical question is whether we can provide explicit bounds for the adaptivity gap for a general class of problems.

Acknowledgements

SJ, HC, and RG were supported by the National Science Foundation (NSF) under award numbers IIS–1939677, OAC–1940224, and IIS–1845434

We extend thanks to Eytan Bakshy and Maximilian Balandat for help with the `BoTorch` package before its release.

References

Dimitri P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific, 2017. ISBN 1-886529-43-4.

Henry Chai and Roman Garnett. Improving quadrature for constrained integrands. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (*AISTATS*)*, 2019.

Henry Chai, Jean-François Ton, Michael A. Osborne, and Roman Garnett. Automated model selection with Bayesian quadrature. In *Proceedings of the 36th International Conference on Machine Learning (*ICML*)*, 2019.

Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923, 2014.

Persi Diaconis. Bayesian numerical analysis. *Statistical Decision Theory and Related Topics*, 4(1):163–175, 1988.

David Ginsbourger and Rodolphe Le Riche. Towards Gaussian process-based optimization with finite time horizon. In *Advances in Model-Oriented Design and Analysis (*MODA*) 9*, pages 89–96, 2010.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. 2010.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (*AISTATS*)*, 2016a.

Javier González, Michael Osborne, and Neil D Lawrence. GLASSES: Relieving the myopia of Bayesian optimisation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (*AISTATS*)*, 2016b.

Tom Gunter, Michael A. Osborne, Roman Garnett, Philipp Hennig, and Stephen J. Roberts. Sampling for inference in probabilistic models with fast Bayesian quadrature. In *Advances in Neural Information Processing Systems (*NEURIPS*) 27*, 2014.

Trong Nghia Hoang, Kian Hsiang Low, Patrick Jaillet, and Mohan Kankanhalli. Nonmyopic ϵ-Bayes-optimal active learning of Gaussian processes. In *Proceedings of the 31st International Conference on Machine Learning (*ICML*)*, 2014.

Shali Jiang, Gustavo Malkomes, Geoff Converse, Alyssa Shofner, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic active search. In *Proceedings of the 34th International Conference on Machine Learning (*ICML*)*, 2017.

Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic batch active search. In *Advances in Neural Information Processing Systems (*NEURIPS*) 31*, 2018.

Donald R. Jones. Direct global optimization algorithm. In *Encyclopedia of optimization*, pages 431–440. 2009.

Andreas Krause and Carlos Guestrin. Nonmyopic active learning of Gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th International Conference on Machine Learning (*ICML*)*, 2007.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

Harold Joseph Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *ASME. J. Basic Eng*, 86(1):97–106, 1964.

Remi Lam, Karen Willcox, and David H. Wolpert. Bayesian optimization with a finite budget: an approximate dynamic programming approach. In *Advances in Neural Information Processing Systems (*NEURIPS*) 29*, 2016.

F. M. Larkin. Gaussian measure in Hilbert space and applications in numerical analysis. *Rocky Mountain Journal of Mathematics*, 2(3):379–422, 1972.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

Chun Kai Ling, Kian Hsiang Low, and Patrick Jaillet. Gaussian process planning with Lipschitz continuous reward functions: towards unifying Bayesian optimization, active learning, and beyond. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Gustavo Malkomes and Roman Garnett. Automating Bayesian optimization with Bayesian optimization. In *Advances in Neural Information Processing Systems (*NEURIPS*) 31*, 2018.

Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1974.

Anthony O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245–260, 1991.

Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian processes for global optimization. In *The 3rd International Conference on Learning and Intelligent Optimization (LION3)*, 2009.

Michael A. Osborne, Roman Garnett, Zoubin Ghahramani, David Duvenaud, Stephen J. Roberts, and Carl E. Rasmussen. Active learning of model evidence using Bayesian quadrature. In *Advances in Neural Information Processing Systems (*NEURIPS*) 25*, 2012.

Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2 edition, 2010.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Burr Settles. Active learning literature survey. Technical report, July 2010.

Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems (*NEURIPS*) 28*, 2015.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: a review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, Jan 2016. ISSN 0018-9219.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (*NEURIPS*) 25*, pages 2951–2959, 2012.

Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning (ICML)*, 2017.

Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems (*NEURIPS*) 29*, 2016.

Jian Wu and Peter Frazier. Practical two-step lookahead Bayesian optimization. In *Advances in Neural Information Processing Systems (*NEURIPS*) 32*, 2019.

Xubo Yue and Raed Al Kontar. Why non-myopic Bayesian optimization is promising and how far should we look-ahead? A study via rollout. *arXiv*, 2019. Accepted to AISTATS 2020.