Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Parallel Bayesian Global Optimization of Expensive Functions

Jialei Wang, Scott C. Clark, Eric Liu, Peter I. Frazier

To cite this article:
Jialei Wang, Scott C. Clark, Eric Liu, Peter I. Frazier (2020) Parallel Bayesian Global Optimization of Expensive Functions. Operations Research 68(6):1850-1865. https://doi.org/10.1287/opre.2019.1966

http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 68, No. 6, November–December 2020, pp. 1850–1865
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

Parallel Bayesian Global Optimization of Expensive Functions

Jialei Wang,[a] Scott C. Clark,[b] Eric Liu,[c] Peter I. Frazier[d]

[a] SensesAI, Beijing 100016, China; [b] SigOpt, San Francisco, California 94104; [c] Yelp, Inc., San Francisco, California 94105; [d] School of Operations Research and Information Engineering, Cornell University, Ithaca, New York 14853

Contact: jialeiwang@senses-ai.com, https://orcid.org/0000-0001-7556-4941 (JW); scott@sigopt.com (SCC); eliu@yelp.com (EL); pf98@cornell.edu, https://orcid.org/0000-0002-3501-3341 (PIF)

Received: March 1, 2016
Revised: September 17, 2017; January 11, 2019
Accepted: September 30, 2019
Published Online in Articles in Advance: June 19, 2020

Subject Classifications: Design of experiments < simulation; Bayesian < statistics; nondifferentiable < programming
Area of Review: Simulation

https://doi.org/10.1287/opre.2019.1966

Abstract. We consider parallel global optimization of derivative-free expensive-to-evaluate functions, and propose an efficient method based on stochastic approximation for implementing a conceptual Bayesian optimization algorithm proposed by Ginsbourger in 2008. At the heart of this algorithm is maximizing the information criterion called the "multipoints expected improvement," or the *q-EI*. To accomplish this, we use infinitesimal perturbation analysis (IPA) to construct a stochastic gradient estimator and show that this estimator is unbiased. We also show that the stochastic gradient ascent algorithm using the constructed gradient estimator converges to a stationary point of the *q-EI* surface, and therefore, as the number of multiple starts of the gradient ascent algorithm and the number of steps for each start grow large, the one-step Bayes-optimal set of points is recovered. We show in numerical experiments using up to 128 parallel evaluations that our method for maximizing the *q-EI* is faster than methods based on closed-form evaluation using high-dimensional integration, when considering many parallel function evaluations, and is comparable in speed when considering few. We also show that the resulting one-step Bayes-optimal algorithm for parallel global optimization finds high-quality solutions with fewer evaluations than a heuristic based on approximately maximizing the *q-EI*. A high-quality open source implementation of this algorithm is available in the open source Metrics Optimization Engine (MOE).

Funding: P. Frazier and J. Wang were partially supported by the National Science Foundation [CAREER CMMI 1254298, CMMI 1536895, and Information and Intelligent Systems 1247696], and the Air Force Office of Scientific Research [FA9550-12-1-0200, FA9550-15-1-0038, and FA9550-16-1-0046].
Supplemental Material: The e-companion is available at https://doi.org/10.1287/opre.2019.1966.

Keywords: Bayesian optimization • parallel optimization • parallel expected improvement • infinitesimal perturbation analysis

1. Introduction

We consider derivative-free global optimization of expensive functions, in which (1) our objective function is time-consuming to evaluate, limiting the number of function evaluations we can perform; (2) evaluating the objective function provides only the value of the objective, and not the gradient or Hessian; and (3) we seek a global, rather than a local, optimum. Such problems arise when the objective function is evaluated by running a complex computer code (see, e.g., Sacks et al. 1989), performing a laboratory experiment, or building a prototype system to be evaluated in the real world. In this paper we assume our function evaluations are deterministic, that is, free from noise.

Bayesian Global Optimization (BGO) methods are one class of methods for solving such problems. They were initially proposed by Kushner (1964), with early work pursued in Mockus et al. (1978) and Mockus (1989), and more recent work including improved algorithms (Boender and Kan 1987, Jones et al. 1998, Huang et al. 2006), convergence analysis (Calvin 1997, Calvin and Žilinskas 2002, Vazquez and Bect 2010), and allowing noisy function evaluations (Calvin and Žilinskas 2005, Huang et al. 2006, Frazier et al. 2009, Villemonteix et al. 2009).

The most well-known BGO method is Efficient Global Optimization (EGO) from Jones et al. (1998), which chooses each point at which to evaluate the expensive objective function in the "outer" expensive global optimization problem by solving an "inner" optimization problem: maximize the "expected improvement." Expected improvement is the value of information (Howard 1966) from a single function evaluation and quantifies the benefit that this evaluation

provides in terms of revealing a point with a better objective function value than previously known. If this is the final point that will be evaluated in the outer optimization problem, and if additional conditions are satisfied (the evaluations are free from noise, and the implementation decision, i.e., the solution that will be implemented in practice after the optimization is complete, is restricted to be a previously evaluated point), then the point with largest expected improvement is the Bayes-optimal point to evaluate, in the sense of providing the best possible average-case performance in the outer expensive global optimization problem (Frazier and Wang 2016).

Solving EGO's inner optimization problem is facilitated by an easy-to-compute and differentiate expression for the expected improvement in terms of the scalar normal cumulative distribution function. Fast evaluation of the expected improvement and its gradient make it possible in many applications to solve the inner optimization problem in significantly less time than the time required per evaluation of the expensive outer objective, which is critical to EGO's usefulness as an optimization algorithm.

The inner optimization problem at the heart of EGO and its objective, the expected improvement, was generalized by Ginsbourger et al. (2008) to the parallel setting, in which the expensive objective can be evaluated at several points simultaneously. This generalization, called the "multipoints expected improvement" or the *q-EI*, is consistent with the decision-theoretic derivation of expected improvement and quantifies the expected utility that will result from the evaluation of a *set* of points. (Here, the increase in utility is the improvement in the objective value.) This work also provided an analytical formula for $q = 2$.

If this generalized inner optimization problem, which is to find the set of points to evaluate next that jointly maximize the *q-EI*, could be solved efficiently, then this would provide the one-step Bayes-optimal set of points to evaluate in the outer problem, and would create a one-step Bayes-optimal algorithm for global optimization of expensive functions able to fully utilize parallelism.

This generalized inner optimization problem is challenging, however, because unlike the scalar expected improvement used by EGO, the *q-EI* lacks an easy-to-compute and differentiate expression, and is calculable only through Monte Carlo simulation, high-dimensional numerical integration, or expressions involving high-dimensional multivariate normal cumulative distribution functions (CDFs). This significantly restricts the set of applications in which a naive implementation can solve the inner problem faster than a single evaluation of the outer optimization problem. Stymied by this difficulty, Ginsbourger et al. (2008) and later work (Chevalier and Ginsbourger 2013) propose heuristic methods that are *motivated* by the one-step optimal algorithm of evaluating the set of points that jointly maximize the *q-EI*, but that do not actually achieve this gold standard.

1.1. Contributions

The main contribution of this work is to provide a method that solves the inner optimization problem of maximizing the *q-EI* efficiently, creating a practical and broadly applicable one-step Bayes-optimal algorithm for parallel global optimization of expensive functions. To accomplish this, we use infinitesimal perturbation analysis (IPA) (Ho 1987) to construct a stochastic gradient estimator of the gradient of the *q-EI* surface, and show that this estimator is unbiased, with a bounded second moment. Our method uses this estimator within a stochastic gradient ascent algorithm, which we show converges to the set of stationary points of the *q-EI* surface. We use multiple restarts to identify multiple stationary points, and then select the best stationary point found. As the number of restarts and the number of iterations of stochastic gradient ascent within each restart both grow large, the one-step optimal set of points to evaluate is recovered.

Our method can be implemented in both synchronous environments, in which function evaluations are performed in batches and finish at the same time, and asynchronous ones, in which a function evaluation may finish before others are done.

In addition to our methodological contribution, we have developed a high-quality open source software package, the "Metrics Optimization Engine (MOE)" (Clark et al. 2014), implementing our method for solving the inner optimization problem and the resulting algorithm for parallel global optimization of expensive functions. To further enhance computational speed, the implementation takes advantage of parallel computing and achieves 100× speedup over single-threaded computation when deployed on a graphical processing unit (GPU). This software package has been used by Yelp and Netflix to solve global optimization problems arising in their businesses (Amatriain 2014, Clark 2014). For the rest of the paper, we refer to our method as "MOE-qEI" because it is implemented in MOE.

We compare MOE-qEI against several benchmark methods. We show that MOE-qEI provides high-quality solutions to the outer optimization problem using fewer function evaluations than the heuristic Constant Liar (CL-mix) policy proposed by Chevalier and Ginsbourger (2013), which is motivated by the inner optimization problem. We also show that MOE-qEI provides a substantial parallel speedup over the single-threaded EGO algorithm, which is one-step optimal when parallel resources are unavailable. We also compare our simulation-based method for solving the inner optimization problem against methods based on exact

evaluation of the *q-EI* from Chevalier and Ginsbourger (2013) and Marmin et al. (2015) (discussed in more detail in this paper) and show that our simulation-based approach to solving the inner optimization problem provides solutions to both the inner and outer optimization problem that are comparable in quality and speed when q is small, and superior when q is large.

1.1.1. Related Work. Developed independently and in parallel with our work is Chevalier and Ginsbourger (2013), which provides a closed-form formula for computing *q-EI*, and the book chapter Marmin et al. (2015), which provides a closed-form expression for its gradient. Both require multiple calls to high-dimensional multivariate normal CDFs. These expressions can be used within an existing continuous optimization algorithm to solve the inner optimization problem that we consider.

Although attractive in that they provide closed-form expressions, calculating these expressions when q is even moderately large is slow and numerically challenging. This is because calculating the multivariate normal CDF in moderately large dimension is itself challenging, with state-of-the-art methods relying on numerical integration or Monte Carlo sampling as described in Genz (1992). Indeed, the method for evaluating the *q-EI* from Chevalier and Ginsbourger (2013) requires q^2 evaluations of the $q-1$ dimensional multivariate normal CDF, and the method for evaluating its gradient requires $O(q^4)$ calls to multivariate normal CDFs with dimension ranging from $q-3$ to q. In our numerical experiments, we demonstrate that our method for solving the inner optimization problem requires less computation time and parallelizes more easily than do these competing methods for $q > 4$, and performs comparably when q is smaller. We also demonstrate that MOE-qEI's improved performance in the inner optimization problem for $q > 4$ translates to improved performance in the outer optimization problem.

Other related work includes the previously proposed heuristic CL-mix from Chevalier and Ginsbourger (2013), which does not solve the inner maximization of *q-EI*, instead using an approximation. Although solving the inner maximization of *q-EI* as we do makes it more expensive to compute the set of points to evaluate next, we show in our numerical experiments that it results in a substantial savings in the number of evaluations required to find a point with a desired quality. When function evaluations are expensive, this results in a substantial reduction in overall time to reach an approximately optimal solution.

In other related work on parallel Bayesian optimization, Frazier et al. (2011) and Xie et al. (2016) proposed a Bayesian optimization algorithm that evaluates pairs of points in parallel, and is one-step Bayes-optimal in the noisy setting under the assumption that one can only observe noisy function values for single points, or noisy function value differences between pairs of points. This algorithm, however, is limited to evaluating pairs of points, and does not extend to a higher level of parallelism.

There are also other non-Bayesian algorithms for derivative-free global optimization of expensive functions with parallel function evaluations from Dennis and Torczon (1991), Kennedy (2010), and Holland (1992). These are quite different in spirit from the algorithm we develop, not being derived from a decision-theoretic foundation.

1.1.2. Outline. We begin in Section 2 by describing the mathematical setting in which Bayesian global optimization is performed, and then defining the *q-EI* and the one-step optimal algorithm. We construct our stochastic gradient estimator in Section 3.2 and use it within stochastic gradient ascent to define a one-step optimal method for parallel Bayesian global optimization in Section 3.3. Then in Section 4.1, we show that the constructed gradient estimator of the *q-EI* surface is unbiased under mild regularity conditions, and in Section 4.2, we provide convergence analysis of the stochastic gradient ascent algorithm. Finally, in Section 5 we present numerical experiments: we compare MOE-qEI against previously proposed heuristics from the literature; we demonstrate that MOE-qEI provides a speedup over single-threaded EGO; we show that MOE-qEI is more efficient than optimizing evaluations of the *q-EI* using closed-form formula provided in Chevalier and Ginsbourger (2013) when q is large; and we show that MOE-qEI computes the gradient of *q-EI* faster than evaluating the closed-form expression proposed in Marmin et al. (2015).

2. Problem Formulation and Background

In this section, we describe a decision-theoretic approach to Bayesian global optimization in parallel computing environments, previously proposed by Ginsbourger et al. (2008). This approach was considered to be purely conceptual as it contains a difficult-to-solve optimization subproblem (our so-called inner optimization problem). In this section, we present this inner optimization problem as background and present a novel method in the subsequent section that solves it efficiently.

2.1. Bayesian Global Optimization

Bayesian global optimization considers optimization of a function f with domain $\mathbb{A} \subseteq \mathbb{R}^d$. The overarching goal is to find an approximate solution to

$$\min_{x \in \mathbb{A}} f(x).$$

We suppose that evaluating f is expensive or time-consuming, and that these evaluations provide only the value of f at the evaluated point and not its gradient or Hessian. We assume that the function defining the domain $\mathbb{A}$ is easy to evaluate and that projections from $\mathbb{R}^d$ into the nearest point in $\mathbb{A}$ can be performed quickly.

Rather than focusing on asymptotic performance as the number of function evaluations grows large, we wish to find an algorithm that performs well, on average, given a limited budget of function evaluations. To formalize this, we model our prior beliefs on the function f with a Bayesian prior distribution, and we suppose that f was drawn at random by nature from this prior distribution, before any evaluations were performed. We then seek to develop an optimization algorithm that will perform well, on average, when applied to a function drawn at random in this way.

2.2. Gaussian Process Priors

For our Bayesian prior distribution on f, we adopt a Gaussian process prior (see Rasmussen and Williams 2006), which is specified by its mean function $\mu(\boldsymbol{x}) : \mathbb{A} \to \mathbb{R}$ and positive semidefinite covariance function $k(\boldsymbol{x}, \boldsymbol{x}') : \mathbb{A} \times \mathbb{A} \to \mathbb{R}$. We write the Gaussian process as

$$f \sim \mathcal{GP}(\mu, k).$$

Then for a collection of points $\boldsymbol{X} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_q)$, the prior of f at $\boldsymbol{X}$ is

$$f(\boldsymbol{X}) \sim \mathcal{N}\left(\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)}\right), \tag{1}$$

where $\boldsymbol{\mu}_i^{(0)} = \mu(\boldsymbol{x}_i)$ and $\boldsymbol{\Sigma}_{ij}^{(0)} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j \in \{1, \ldots, q\}$.

Our proposed method for choosing the points to evaluate next additionally requires that μ and k satisfy some mild regularity assumptions discussed later in this paper, but otherwise adds no additional requirements. In practice, μ and k are typically chosen using an empirical Bayes approach discussed in Brochu et al. (2010), in which first, a parameterized functional form for μ and k is assumed; second, a first stage of data are collected in which f is evaluated at points chosen according to a Latin hypercube or uniform design; and third, maximum likelihood estimates for the parameters specifying m and k are obtained. In some implementations (Jones et al. 1998, Snoek et al. 2012), these estimates are updated iteratively as more evaluations of f are obtained, which provides more accurate inference and tends to reduce the number of function evaluations required to find good solutions but increases the computational overhead per evaluation. We adopt this method in our numerical experiments in Section 5. However, the specific contribution of this paper, a new method for solving an optimization subproblem arising in the choice of design points, works with any choice of mean function μ and covariance matrix k, as long as they satisfy the mild regularity conditions discussed later in this paper.

In addition to the prior distribution specified in (1), we may also have some previously observed function values $y^{(i)} = f(\boldsymbol{x}^{(i)})$, for $i = 1, \ldots, n$. These might have been obtained through the previously mentioned first stage of sampling, running the second stage sampling method we are about to describe, or from some additional runs of the expensive objective function f performed by another party outside of the control of our algorithm. If no additional function values are available, we set $n = 0$. We define notation $\boldsymbol{x}^{(1:n)} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)})$ and $y^{(1:n)} = (y^{(1)}, \ldots, y^{(n)})$. We require that all points in $\boldsymbol{x}^{(1:n)}$ be distinct.

We then combine these previously observed function values with our prior one to obtain a posterior distribution on $f(\boldsymbol{X})$. This posterior distribution is still a multivariate normal (e.g., see equation (A.6) on p. 200 in Rasmussen and Williams 2006)

$$f(\boldsymbol{X}) \mid \boldsymbol{X}, \boldsymbol{x}^{(1:n)}, y^{(1:n)} \sim \mathcal{N}\left(\boldsymbol{\mu}^{(n)}, \boldsymbol{\Sigma}^{(n)}\right), \tag{2}$$

with

$$\begin{aligned} \boldsymbol{\mu}^{(n)} &= \boldsymbol{\mu}^{(0)} + K\left(\boldsymbol{X}, \boldsymbol{x}^{(1:n)}\right) K\left(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)}\right)^{-1} \\ &\quad \times \left(y^{(1:n)} - \mu\left(\boldsymbol{x}^{(1:n)}\right)\right), \\ \boldsymbol{\Sigma}^{(n)} &= K(\boldsymbol{X}, \boldsymbol{X}) - K\left(\boldsymbol{X}, \boldsymbol{x}^{(1:n)}\right) K\left(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)}\right)^{-1} \\ &\quad \times K\left(\boldsymbol{x}^{(1:n)}, \boldsymbol{X}\right), \end{aligned} \tag{3}$$

where $\mu(\boldsymbol{x}^{(1:n)})$ is the vector obtained by evaluating the prior mean function at each point in $\boldsymbol{x}^{(1:n)}$, $K(\boldsymbol{X}, \boldsymbol{x}^{(1:n)})$ is a $q \times n$ matrix with $K(\boldsymbol{X}, \boldsymbol{x}^{(1:n)})_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}^{(j)})$, and similarly for $K(\boldsymbol{x}^{(1:n)}, \boldsymbol{X})$, $K(\boldsymbol{X}, \boldsymbol{X})$ and $K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)})$.

2.3. Multipoints Expected Improvement (*q-EI*)

In a parallel computing environment, we wish to use this posterior distribution to choose the set of points to evaluate next. Ginsbourger et al. (2008) proposed making this choice using a decision-theoretic approach that considers the utility provided by evaluating a particular candidate set of points in terms of their ability to reveal better solutions than previously known. We review this decision-theoretic approach here and then present a new algorithm for implementing this choice in the next section.

Let q be the number of function evaluations we will perform in parallel, and let $\boldsymbol{X}$ be a candidate set of points that we are considering evaluating next. Let $f_n^\star = \min_{m \le n} f(\boldsymbol{x}^{(m)})$ indicate the value of the best point evaluated, before beginning these q new function evaluations. The value of the best point evaluated after all q function evaluations are complete will be $\min(f_n^\star, \min_{i=1,\ldots,q} f(\boldsymbol{x}_i))$. The difference between these two values (the values of the best point evaluated,

before and after these q new function evaluations) is called the *improvement*, and is equal to $(f_n^\star - \min_{i=1,\dots,q} f(\boldsymbol{x}_i))^+$, where $a^+ = \max(a, 0)$ for $a \in \mathbb{R}$.

We then compute the expectation of this improvement over the joint probability distribution over $f(\boldsymbol{x}_i)$, $i = 1, \dots, q$, and we refer to this quantity as the *multipoints expected improvement* or *q-EI* from Ginsbourger et al. (2008). This multipoints expected improvement can be written as

$$q\text{-}EI(\boldsymbol{X}) = \mathbb{E}_n\left[\left(f_n^\star - \min_{i=1,\dots,q} f(\boldsymbol{x}_i)\right)^+\right], \tag{4}$$

where $\mathbb{E}_n[\cdot] := \mathbb{E}[\cdot|\boldsymbol{x}^{(1:n)}, y^{(1:n)}]$ is the expectation taken with respect to the posterior distribution.

Ginsbourger et al. (2008) then proposes evaluating next the set of points that maximize the multipoints expected improvement,

$$\arg\max_{X \in H} q\text{-}EI(\boldsymbol{X}), \tag{5}$$

where $H = \{(\boldsymbol{x}_1, \dots, \boldsymbol{x}_q) : \boldsymbol{x}_i \in \mathbb{A}, \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq r, \|\boldsymbol{x}_i - \boldsymbol{x}^{(\ell)}\| \geq r, i \neq j, 1 \leq i, j \leq q, 1 \leq \ell \leq n\}$.

This formulation generalizes Ginsbourger et al. (2008) slightly by allowing an optional requirement that new evaluation points be a distance of at least $r \geq 0$ from each other and previously evaluated points. Ginsbourger et al. (2008) implicitly set $r = 0$. Our convergence proof requires $r > 0$, which provides a compact feasible domain over which the stochastic gradient estimator has bounded variance. Setting a strictly positive r can also improve numerical stability in inference (see, e.g., Ababou et al. 1994), and evaluating a point extremely close to a previously evaluated point is typically unlikely to provide substantial improvement in the revealed objective value. In our experiments we set $r = 10^{-5}$.

In the special case $q = 1$, which occurs when we are operating without parallelism, the multipoints expected improvement reduces to the expected improvement (Mockus 1989, Jones et al. 1998), which can be evaluated in closed-form in terms of the normal density and CDF as discussed in Section 1. Ginsbourger et al. (2008) provided an analytical expression for *q-EI* when $q = 2$, but in the same paper the authors commented that computing *q-EI* for $q > 2$ involves expensive-to-compute q-dimensional Gaussian cumulative distribution functions relying on multivariate integral approximation, which makes solving (5) difficult. Ginsbourger (2009, p. 1) writes "directly optimizing the *q-EI* becomes extremely expensive as q and d (the dimension of inputs) grow."

3. Algorithm

In this section, we present a new algorithm for solving the inner optimization problem (5) of maximizing *q-EI*. This algorithm uses a novel estimator of the gradient of the *q-EI* presented in Section 3.2, used within a multistart stochastic gradient ascent framework as described in Section 3.3. We additionally generalize this technique from synchronous to asynchronous parallel optimization in Section 3.4. Section 3.1 begins by introducing additional notation used to describe our algorithm.

Although we keep q fixed (typically to the maximum level of parallelism in one's computing environment), there are settings where one may wish to choose it in a more refined way: if parallelism can be flexibly purchased in a cloud computing environment; or if parallel resources can be devoted to parallelizing each function evaluation in addition to running multiple function evaluations in parallel. Section 1.2 in the e-companion briefly discusses choosing q in these settings.

3.1. Notation

In this section, we define additional notation to better support construction of the gradient estimator. Justified by (2), we write $f(\boldsymbol{X})$ as

$$f(\boldsymbol{X}) \overset{d}{=} \boldsymbol{\mu}(\boldsymbol{X}) + \boldsymbol{L}(\boldsymbol{X})\boldsymbol{Z}, \tag{6}$$

where $\boldsymbol{L}(\boldsymbol{X})$ is the lower triangular matrix obtained from the Cholesky decomposition of $\boldsymbol{\Sigma}^{(n)}$ in (2), $\boldsymbol{\mu}(\boldsymbol{X})$ is the posterior mean (identical to $\boldsymbol{\mu}^{(n)}$ in (2), but rewritten here to emphasize the dependence on X and de-emphasize the dependence on n), and $\boldsymbol{Z}$ is a multivariate standard normal random vector. We will also use the notation $\boldsymbol{\Sigma}(\boldsymbol{X})$ in place of $\boldsymbol{\Sigma}^{(n)}$ in our analysis.

By substituting (6) into (4), we have

$$q\text{-}EI(\boldsymbol{X}) = \mathbb{E}\left[\left(f_n^* - \min_{i=1,\dots,q} \boldsymbol{e}_i\left[\boldsymbol{\mu}(\boldsymbol{X}) + \boldsymbol{L}(\boldsymbol{X})\boldsymbol{Z}\right]\right)^+\right], \tag{7}$$

where $\boldsymbol{e}_i$ is a unit vector in direction i and the expectation is over $\boldsymbol{Z}$. To make (7) even more compact, define a new vector $\boldsymbol{m}(\boldsymbol{X})$ and new matrix $\boldsymbol{C}(\boldsymbol{X})$,

$$\boldsymbol{m}(\boldsymbol{X})_i = \begin{cases} f_n^* - \boldsymbol{\mu}(\boldsymbol{X})_i & \text{if } i > 0, \\ 0 & \text{if } i = 0, \end{cases}$$
$$\boldsymbol{C}(\boldsymbol{X})_{ij} = \begin{cases} -\boldsymbol{L}(\boldsymbol{X})_{ij} & \text{if } i > 0, \\ 0 & \text{if } i = 0, \end{cases} \tag{8}$$

and (7) becomes

$$q\text{-}EI(\boldsymbol{X}) = \mathbb{E}\left[\max_{i=0,\dots,q} \boldsymbol{e}_i[\boldsymbol{m}(\boldsymbol{X}) + \boldsymbol{C}(\boldsymbol{X})\boldsymbol{Z}]\right]. \tag{9}$$

3.2. Constructing the Gradient Estimator

We now construct our estimator of the gradient $\nabla q\text{-}EI(\boldsymbol{X})$. Let

$$h(\boldsymbol{X}, \boldsymbol{Z}) = \max_{i=0,\dots,q} \boldsymbol{e}_i[\boldsymbol{m}(\boldsymbol{X}) + \boldsymbol{C}(\boldsymbol{X})\boldsymbol{Z}]. \tag{10}$$

Then

$$\nabla q\text{-}EI(\boldsymbol{X}) = \nabla \mathbb{E} h(\boldsymbol{X}, \boldsymbol{Z}). \tag{11}$$

If gradient and expectation in (11) are interchangeable, the gradient would be

$$\nabla q\text{-}EI(\boldsymbol{X}) = \mathbb{E} \boldsymbol{g}(\boldsymbol{X}, \boldsymbol{Z}), \tag{12}$$

where

$$\boldsymbol{g}(\boldsymbol{X}, \boldsymbol{Z}) = \begin{cases} \nabla h(\boldsymbol{X}, \boldsymbol{Z}) & \text{if } \nabla h(\boldsymbol{X}, \boldsymbol{Z}) \text{ exists,} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

$\boldsymbol{g}(\boldsymbol{X}, \boldsymbol{Z})$ can be computed using results on differentiation of the Cholesky decomposition from Smith (1995).

We use $\boldsymbol{g}(\boldsymbol{X}, \boldsymbol{Z})$ as our estimator of the gradient $\nabla q\text{-}EI$ and will discuss interchangeability of gradient and expectation, which implies unbiasedness of our gradient estimator, in Section 4.1. As will be discussed in Section 4.2, unbiasedness of the gradient estimator is one of the sufficient conditions for convergence of the stochastic gradient ascent algorithm proposed in Section 3.3.

3.3. Optimization of *q-EI*

Our stochastic gradient ascent algorithm begins with some initial point $\boldsymbol{X}_0 \in H$ and generates a sequence $\{\boldsymbol{X}_t : t = 1, 2, \ldots\}$ using

$$\boldsymbol{X}_{t+1} = \prod_H [\boldsymbol{X}_t + \epsilon_t \boldsymbol{G}(\boldsymbol{X}_t)], \tag{14}$$

where $\prod_H(\boldsymbol{X})$ denotes the closest point in H to $\boldsymbol{X}$, and if the closest point is not unique, a closest point such that the function $\prod_H(\cdot)$ is measurable. $\boldsymbol{G}(\boldsymbol{X}_t)$ is an estimate of the gradient of $q\text{-}EI(\cdot)$ at $\boldsymbol{X}_t$, obtained by averaging M replicates of our stochastic gradient estimator,

$$\boldsymbol{G}(\boldsymbol{X}_t) = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{g}(\boldsymbol{X}_t, \boldsymbol{Z}_{t,m}), \tag{15}$$

where $\{\boldsymbol{Z}_{t,m}$: m=1, . . . , M$\}$ are i.i.d. samples generated from the multivariate standard normal distribution, $\boldsymbol{g}(\boldsymbol{X}_t, \boldsymbol{Z}_{t,m})$ is defined in (13). $\{\epsilon_t : t = 0, 1, \ldots\}$ is a stochastic gradient stepsize sequence (Kushner and Yin 2003), typically chosen to be equal to $\epsilon_t = \frac{a}{t^\gamma}$ for some scalar a and $\gamma \in (0, 1]$. Because we use Polyak-Ruppert averaging as described later in this paper, we set $\gamma < 1$. Analysis in Section 4 shows that, under certain mild conditions, this stochastic gradient algorithm converges almost surely to the set of stationary points.

After running T iterations of stochastic gradient ascent using (14), we obtain the sequence $\{\boldsymbol{X}_t : t = 1, 2, \ldots, T\}$. From this sequence we extract the average $\overline{X}_T = \frac{1}{T+1} \sum_{t=0}^{T} \boldsymbol{X}_t$ and use it as an estimated stationary point. This Polyak-Ruppert averaging approach (Ruppert 1988, Polyak 1990) is more robust to misspecification of the stepsize sequence than using $\boldsymbol{X}_T$ directly.

To find the global maximum of the *q-EI*, we use multiple restarts of the algorithm from a set of starting points, drawn from a Latin hypercube design (McKay et al. 2000), to find multiple stationary points, and then use simulation to evaluate *q-EI* at these stationary points and select the point for which it is largest. For simplicity we present our approach using a fixed sample size N to perform this evaluation and selection (Step 8 in Algorithm 1) but one could also use a more sophisticated ranking and selection algorithm with adaptive sample sizes (see, e.g., Kim and Nelson 2007), or evaluate *q-EI* using the closed-form formula in Chevalier and Ginsbourger (2013). We summarize our procedure for selecting the set of points to sample next, which we call MOE-qEI, in Algorithm 1.

Algorithm 1 MOE-qEI: Optimization of *q-EI*

Require: number of starting points R; stepsize constants a and γ; number of steps for one run of gradient ascent T; number of Monte Carlo samples for estimating the gradient M; number of Monte Carlo samples for estimating *q-EI* N.

1. Draw R starting points from a Latin hypercube design in H, $\boldsymbol{X}_{r,0}$ for $r = 1, \ldots, R$.
2. **For** $r = 1$ to R **do**
3. **For** $t = 0$ to $T - 1$ **do**
4. Compute $\boldsymbol{G}_t = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{g}(\boldsymbol{X}_{r,t}, \boldsymbol{Z}_{r,t,m})$ where $\boldsymbol{Z}_{r,t,m}$ is a vector of q i.i.d. samples drawn from the standard normal distribution.
5. Update solution using stochastic gradient ascent $\boldsymbol{X}_{r,t+1} = \prod_H [\boldsymbol{X}_{r,t} + \frac{a}{t^\gamma} \boldsymbol{G}_t]$.
6. **End for**
7. Compute the simple average of the solutions for $\boldsymbol{X}_{r,t}$, $\overline{\boldsymbol{X}}_{r,T} = \frac{1}{T+1} \sum_{t=0}^{T} \boldsymbol{X}_{r,t}$.
8. Estimate $q\text{-}EI(\overline{\boldsymbol{X}}_{r,T})$ using Monte Carlo simulation with N i.i.d. samples, and store the estimate as $\widehat{q\text{-}EI}_r$.
9. **End for**
10. **Return** $\overline{\boldsymbol{X}}_{r',T}$ where $r' = \arg\max_{r=1,\ldots,R} \widehat{q\text{-}EI}_r$.

The MOE software package (Clark et al. 2014) implements Algorithm 1 and supplies the following additional optional fallback logic. If $\max_{r=1,\ldots,R} \widehat{q\text{-}EI}_r \le \epsilon'$, so that multistart stochastic gradient ascent fails to find a point with estimated expected improvement better than ϵ', then it generates L additional solutions from a Latin Hypercube on H, estimates the *q-EI* at each of these using the same Monte Carlo approach as in Step 8, and selects the one with the largest estimated *q-EI*. This logic takes two additional parameters: a strictly positive real number ϵ' and an integer L. We turn this logic off in our experiments by setting $\epsilon' = 0$.

3.4. Asynchronous Parallel Optimization

So far we have assumed synchronous parallel optimization, in which we wait for all q points to finish before choosing a new set of points. However, in some applications, we may wish to generate a new partial batch of points to evaluate next while p points are still being evaluated, before we have their values. This is common in expensive computer evaluations, which do not necessarily finish at the same time.

We can extend Algorithm 1 to solve an extension of (5) proposed by Ginsbourger et al. (2010) for asynchronous parallel optimization: suppose parallelization allows a batch of q points to be evaluated simultaneously; the first p points are still under evaluation, whereas the remaining $q-p$ points have finished evaluation and the resources used to evaluate them are free to evaluate new points. We let $\boldsymbol{X}' := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ be the first p points still under evaluation and let $\boldsymbol{X} := (\boldsymbol{x}_{p+1}, \ldots, \boldsymbol{x}_q)$ be the $(q-p)$ points ready for new evaluations. Computation of q-*EI* for these q points remains the same as in (4), but we use an alternative notation, $q\text{-}EI(\boldsymbol{X}', \boldsymbol{X})$, to explicitly indicate that $\boldsymbol{X}'$ are the points still being evaluated and $\boldsymbol{X}$ are the new points to evaluate. We emphasize that the expectation in $q\text{-}EI(\boldsymbol{X}', \boldsymbol{X})$ is over yet-to-be-observed values of f at both $\mathbf{X}$ and $\mathbf{X}'$, and is taken with respect to the posterior given the previous n evaluations. Keeping $\boldsymbol{X}'$ fixed, we optimize q-*EI* over $\boldsymbol{X}$ by solving this alternative problem

$$\arg\max_{\boldsymbol{X} \in H'} q\text{-}EI(\boldsymbol{X}', \boldsymbol{X}), \tag{16}$$

where $H' = \{(\boldsymbol{x}_{p+1}, \ldots, \boldsymbol{x}_q) : \boldsymbol{x}_i \in \mathbb{A}, \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq r, \|\boldsymbol{x}_i - \boldsymbol{x}_k\| \geq r, \|\boldsymbol{x}_i - \boldsymbol{x}^{(m)}\| \geq r, i \neq j, p < i \leq q, p < j \leq q, 1 \leq k \leq p, 1 \leq m \leq n\}$ for some small positive r. As we did in the algorithm for synchronous parallel optimization in Section 3.3, we estimate the gradient of the objective function with respect to $\boldsymbol{X}$, that is, $\nabla_{\boldsymbol{X}} q\text{-}EI(\boldsymbol{X}', \boldsymbol{X})$. The gradient estimator is essentially the same as that in Section 3.2, except that we only differentiate $h(\cdot, \cdot)$ with respect to $\boldsymbol{X}$. (Although $h(\cdot, \cdot)$ is only differentiated with respect to $\mathbf{X}$, it depends on both $\mathbf{X}$ and $\mathbf{X}'$.) Then we proceed according to Algorithm 1.

In practice, one typically sets $p = q - 1$. This is because Bayesian optimization procedures are used most frequently when function evaluation times are large, and asynchronous computing environments typically have a time between evaluation completions that increases with the evaluation time. When this intercompletion time is large relative to the time required to solve (16), it is typically better to solve (16) each time an evaluation completes, that is, to set $p = q - 1$.

Indeed, if we set $p < q - 1$, then we let CPU cores sit idle for longer and we decrease the total utilization of our parallel computing environment. For example, consider the CPU core that finishes first. When $p = q - 1$, it waits to start a new function evaluation only for the time it takes to maximize q-*EI*. Moreover, this core can be used to do this maximization so that no time is wasted. But, if $p < q - 1$, it must sit idle while we wait for an additional $q - 1 - p$ cores to complete. By not letting cores sit idle, we reduce the wall-clock time required to do a given number of function evaluations.

If the time to perform a function evaluation is small enough, or if the computing environment is especially homogeneous, then the time between completions might be substantially smaller than the time to solve (16) and one might wish to set p strictly smaller than $q - 1$. This may be beneficial because short intercompletion times keep the resulting increase in idle time small and it allows results from a larger group of function evaluations to be available when deciding how to next allocate cores.

4. Theoretical Analysis

In Section 3, when we constructed our gradient estimator and described the use of stochastic gradient ascent to optimize q-*EI*, we alluded to conditions under which this gradient estimator is unbiased and this stochastic gradient ascent algorithm converges to the set of stationary points of the q-*EI* surface. In this section, we describe these conditions and state these results.

4.1. Unbiasedness of the Gradient Estimator

We now state our main theorem showing unbiasedness of the gradient estimator. Proofs of all results including supporting lemmas are available as supplemental material.

Theorem 1. *If $\boldsymbol{m}(\boldsymbol{X})$ and $\boldsymbol{C}(\boldsymbol{X})$ are continuously differentiable in a neighborhood of $\boldsymbol{X}$, and $\boldsymbol{C}(\boldsymbol{X})$ has no duplicate rows, then $\nabla h(\boldsymbol{X}, \boldsymbol{Z})$ exists almost surely and*

$$\nabla \mathbb{E} h(\boldsymbol{X}, \boldsymbol{Z}) = \mathbb{E} \nabla h(\boldsymbol{X}, \boldsymbol{Z}).$$

Theorem 1 requires continuous differentiability of $\boldsymbol{C}(\boldsymbol{X})$, which may seem difficult to verify. However, using Smith (1995), which shows that mth-order differentiability of a symmetric and nonnegative definite matrix implies mth-order differentiability of the lower triangular matrix obtained from its Cholesky factorization, $\boldsymbol{L}(\boldsymbol{X})$ and thus $\boldsymbol{C}(\boldsymbol{X})$ have the same order of differentiability as $\boldsymbol{\Sigma}^{(n)}$, whose order of differentiability can in turn be verified by examination of the prior covariance function $k(\cdot, \cdot)$. In addition, when $\boldsymbol{\Sigma}^{(n)}$ is positive definite, $\boldsymbol{C}(\boldsymbol{X})$ will not have duplicate rows. We will use these facts in Corollary 1, after first discussing convergence, to provide easy-to-verify conditions under which unbiasedness and convergence to the set of stationary points hold.

4.2. Convergence Analysis

In this section, we show almost sure convergence of our proposed stochastic gradient ascent algorithm. We assume that $\mathbb{A}$ is compact and can be written in the form $\mathbb{A} = \{\boldsymbol{x} : a_i'(\boldsymbol{x}) \le 0, i = 1, \ldots, m'\} \subseteq \mathbb{R}^d$, where $a_i'(\cdot)$ is any real-valued constraint function. Then H can be written in a form more convenient for analysis,

$$H = \{\boldsymbol{X} : a_i(\boldsymbol{X}) \le 0, i = 1, \ldots, m\} \subseteq \mathbb{R}^{d \times q},$$

where $a_{(i-1)q+j}(\boldsymbol{X}) = a_i'(\boldsymbol{x}_j)$ with $\boldsymbol{x}_j$ being the jth point in $\boldsymbol{X}$, and $a_i(\boldsymbol{X})$ for $i > m'q$ encodes the constraints $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \ge r$ and $\|\boldsymbol{x}_i - \boldsymbol{x}^{(\ell)}\| \ge r$ present in (5).

The following theorem shows that Algorithm 1 converges to the set of stationary points under conditions that include those of Theorem 1. The proof is available as supplemental material.

Theorem 2. *Suppose the following assumptions hold,*

1. $a_i(\cdot), i = 1, \ldots, m$ *are continuously differentiable.*
2. $\epsilon_t \to 0$ *for* $t \ge 0$; $\sum_{t=1}^{\infty} \epsilon_t = \infty$ *and* $\sum_{t=0}^{\infty} \epsilon_t^2 < \infty$.
3. $\forall \boldsymbol{X} \in H$, $\boldsymbol{\mu}(\boldsymbol{X})$, *and* $\boldsymbol{\Sigma}(\boldsymbol{X})$ *are twice continuously differentiable, and* $\boldsymbol{\Sigma}(\boldsymbol{X})$ *is positive definite.*

Then the sequence $\{\boldsymbol{X}_t : t = 0, 1, \ldots\}$ *and its Polyak-Ruppert average* $\{\overline{\boldsymbol{X}}_t : t = 0, 1, \ldots\}$ *generated by Algorithm* (14) *converges almost surely to a connected set of stationary points of the q-EI surface.*

The following corollary of Theorem 2 uses conditions that can be more easily checked prior to running MOE-qEI. It requires that the sampled points are distinct, which can be made true by dropping duplicate samples. Because function evaluations are deterministic, no information is lost in doing so.

Corollary 1. *If the sampled points* $\boldsymbol{x}^{(1:n)}$ *are distinct and*

1. *the prior covariance function k is positive definite and twice differentiable,*
2. *the prior mean function* μ *is twice differentiable,*
3. *conditions* 1 *and* 2 *in Theorem 2 are met,*

then $\boldsymbol{X}_t$ *and its Polyak-Ruppert average* $\overline{\boldsymbol{X}}_t$ *converge to a connected set of stationary points.*

Proof to Corollary 1. Because $\boldsymbol{x}^{(1:n)}$ are distinct, and $\boldsymbol{X} \in H$, and the prior covariance function is positive definite and twice continuously differentiable, then $K(\boldsymbol{X}, \boldsymbol{x}^{(1:n)})$, $K(\boldsymbol{x}^{(1:n)}, \boldsymbol{X})$, $K(\boldsymbol{X}, \boldsymbol{X})$, and $K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)})$ in (3) are all positive definite and twice continuously differentiable. Because the prior mean function is also twice continuously differentiable, it follows that $\boldsymbol{\mu}^{(n)} = \boldsymbol{\mu}(\boldsymbol{X})$ and $\boldsymbol{\Sigma}^{(n)} = \boldsymbol{\Sigma}(\boldsymbol{X})$ defined in (3) are twice continuously differentiable, and in addition, $\boldsymbol{\Sigma}^{(n)}$ is positive definite. Thus, the conditions of Theorem 2 are verified, and its conclusion holds. □

5. Numerical Results

In this section, we present numerical experiments demonstrating the performance of MOE-qEI. The implementation of MOE-qEI follows Algorithm 1 and is available in the open source software package "MOE" (Clark et al. 2014).

We first discuss the choice of constants in Algorithm 1: R, T, M, N, γ, and a.

1. Number of starting points, R: This should be larger and of the same order as the number of equivalence classes of stationary points of the *q-EI* surface, where we identify a set of stationary points as in the same class if they can be obtained from each other by permuting $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_q$. (*q-EI* is symmetric and such permutations do not change its value.) However, we do not know the number of such equivalence classes, and their number tends to grow with n as the surface grows more modes. Setting R larger increases our chance of finding the global maximum but increases computation. In our numerical experiments, we set $R = n$ to capture this trade-off between runtime and solution quality. As a diagnostic, one can check whether R is large enough by checking the number of unique solutions we obtain; if we obtain the same solution repeatedly from multiple restarts, this suggests R is large enough.

2. Number of steps in stochastic gradient ascent, T, and stepsize sequence parameters a and γ: For simplicity, we set $a = 1$. We set $\gamma = 0.7$, which is significantly below 1, to ensure that the stepsize sequence decreases slowly, as is recommended when using Polyak-Ruppert averaging. We then plotted the *q-EI* and norm of the gradient from stochastic gradient ascent versus t for a few sample problems. Finding that convergence occurred well before the 100th iterate, we set $T = 100$. As a diagnostic, one may also assess convergence by evaluating the gradient using a large number of Monte Carlo samples at the final iterate T and comparing its norm to 0.

3. Number of Monte Carlo samples M: This determines the accuracy of the gradient estimate and therefore affects stochastic gradient ascent's convergence. We set $M = 1000$ and as discussed in Section 3.3 performed an experiment to justify this setting. Although we ran our experiments on a CPU except where otherwise stated to ensure a fair comparison with other competing algorithms, for which a GPU implementation is not available, the MOE software package provides a GPU implementation that can be used to increase the amount of parallelism used in MOE-qEI. When using the GPU implementation, we recommend setting $M = 10^6$ because the GPU's parallelism makes averaging a large number of independent replicates fast, and the reduction in noise reduces the number of iterates needed for convergence by stochastic gradient ascent.

4. Number of Monte Carlo samples for estimating *q-EI*, N: We estimate the *q-EI* at a limiting solution only once for each restart, that is, R times, and so

setting N large introduces little computational overhead. We set $N = 10^6$ to ensure an essentially noise-free selection of the best of the limiting solutions, and we assess this choice by examining the standard error of our estimates of the *q-EI*.

For the outer optimization of the objective function, we begin with a small data set, typically sampled using a Latin hypercube design, to train the Gaussian Process model described in Section 2.2. In our numerical experiments, we use $\mu = 0$ and a squared exponential kernel k whose hyperparameters are estimated using an empirical Bayes approach: we set them to the values that maximize the log marginal likelihood of the observed data. With the trained Gaussian Process model, we perform the inner optimization of MOE-qEI described in Algorithm 1 to find the batch of points to evaluate, and after evaluating them we update the hyperparameters as well as the Gaussian Process model. We repeat this process over a number of iterations and report the best solution found in each iteration.

Noise-free function evaluations may often lead to ill-conditioned covariance matrices $K(\cdot,\cdot)$ in (2). To resolve this problem, we adopt a standard trick from Gaussian process regression (Rasmussen and Williams 2006, section 3.4.3): we manually impose a small amount of noise $\sim \mathcal{N}(0,\sigma^2)$ where $\sigma^2 = 10^{-4}$ and use Gaussian Process regression designed for noisy settings, which is almost identical to (2) except that $K(\boldsymbol{x}^{(1:n)},\boldsymbol{x}^{(1:n)})$ is replaced by $K(\boldsymbol{x}^{(1:n)},\boldsymbol{x}^{(1:n)}) + \sigma^2 I_n$ where I_n is the identity matrix (Rasmussen and Williams 2006, Section 2.2).

5.1. Comparison on the Outer Optimization Problem

Constant Liar is a heuristic algorithm motivated by (9) proposed by Ginsbourger et al. (2008), which uses a greedy approach to iteratively construct a batch of q points. At each iteration of this greedy approach, the heuristic uses the sequential EGO algorithm to find a point that maximizes the expected improvement. However, because the posterior used by EGO depends on the current batch of points, which have not yet been evaluated, Constant Liar imposes a heuristic response (the "liar") at this point, and updates the Gaussian Process model with this liar value. The algorithm stops when q points are added, and reports the batch for function evaluation.

There are three variants of Constant Liar (CL), which use three different strategies for choosing the liar value: CL-min sets the liar value to the minimum response observed so far; CL-max sets it to the maximum response observed so far; and CL-mix is a hybrid of the two, computing one set of points using CL-min, another set of points using CL-max, and sampling the set that has the higher *q-EI*. Among the three methods, CL-mix was shown by Chevalier and Ginsbourger (2013) to have the best overall performance, and therefore we compare MOE-qEI against CL-mix.

We ran MOE-qEI and CL-mix on a range of standard test functions for global optimization (Jamil and Yang 2013): 2-dimensional Branin2; 3-dimensional Hartmann3; 5-dimensional Ackley5; and 6-dimensional Hartmann6. In the experiment, we first draw $(2d + 2)$ points in the domain using a Latin hypercube design, where d is the dimension of the objective function, and fit a Gaussian Process model using the initial points. Thereafter, we let MOE-qEI and CL-mix optimize over the test functions and report for each iteration the regret as regret = f^* − best solution so far for each iteration, where we note that each of the problems considered is a minimization problem. We repeat the experiment 100 times using different initial sets of points, and report the average performance of both algorithms in Figure 1. The result shows that MOE-qEI consistently finds better solutions than the heuristic method on all four test functions.

Next, we compare MOE-qEI and CL-MIX at different levels of parallelism using the same experimental setup. The sequential EGO algorithm makes the same decisions as MOE-qEI when $q = 1$ and so this may also be seen as a comparison against EGO. Figure 2 shows that MOE-qEI achieves significant speedup over EGO as q grows, indicating substantial potential time saving using parallelization and MOE-qEI in Bayesian optimization tasks.

5.2. Comparison on the Inner Optimization Problem

Chevalier and Ginsbourger (2013, p. 1) provided a closed-form formula for *q-EI* and argued that it computes *q-EI* "very fast for reasonably low values of q (typically less than 10)." The closed-form formula is provided as follows for reference, modified to use the notation in this paper. Recall (2) and let $f(\boldsymbol{X}) = (Y_1,\dots,Y_q)$ be a random vector with mean $\boldsymbol{\mu}^{(n)}$ and covariance matrix $\boldsymbol{\Sigma}^{(n)}$. For $k \in \{1,\dots,q\}$, consider the vectors $\boldsymbol{Z}^k := (Z_1^k,\dots,Z_q^k)$ defined as follows:

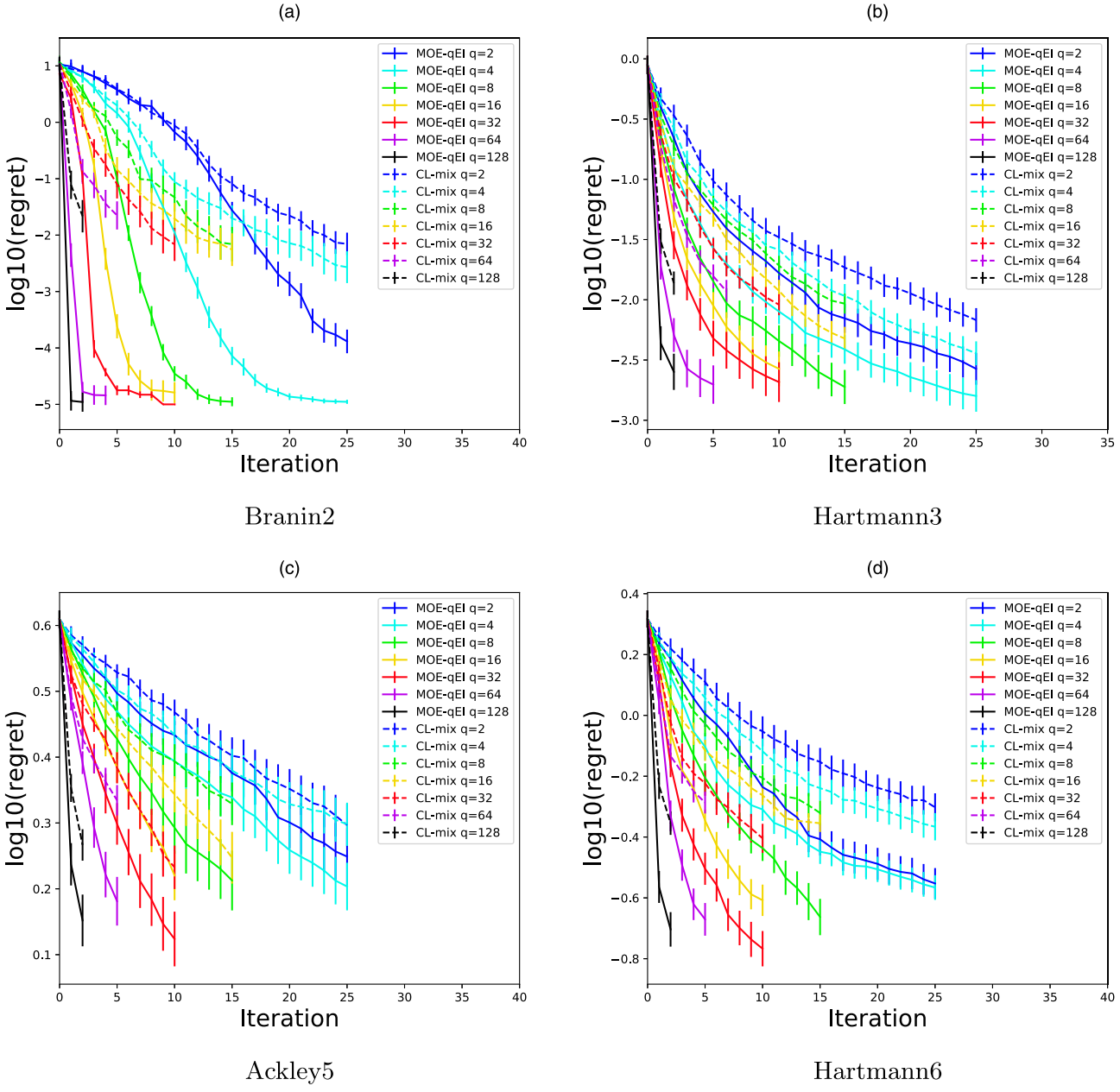
$$Z_j^k := Y_k - Y_j, j \neq k, Z_k^k := Y_k.$$

Let $\boldsymbol{m}^k$ and Σ^k denote the mean and covariance matrix of $\boldsymbol{Z}^k$, and define the vector $\boldsymbol{b}^k \in \mathbb{R}^q$ by $b_k^k = f_n^*$ and $b_j^k = 0$ if $j \neq k$. Then the closed-form formula is

$$\begin{aligned} q\text{-}EI(\boldsymbol{X}) = \sum_{k=1}^{q} \Bigg(& \left(f_n^* - \mu_k^{(n)}\right) \Phi_q\left(\boldsymbol{b}^k - \boldsymbol{m}^k, \Sigma^k\right) \\ & + \sum_{i=1}^{q} \Sigma_{ik}^k \phi_{m_i^k,\Sigma_{ii}^k}\left(b_i^{(k)}\right) \Phi_{q-1}\left(\boldsymbol{c}_{.i}^k, \Sigma_{.i}^k\right) \Bigg), \end{aligned} \tag{17}$$

where $\boldsymbol{c}^k$ is as defined in Chevalier and Ginsbourger (2013).

Figure 1. Comparison Against Constant Liar

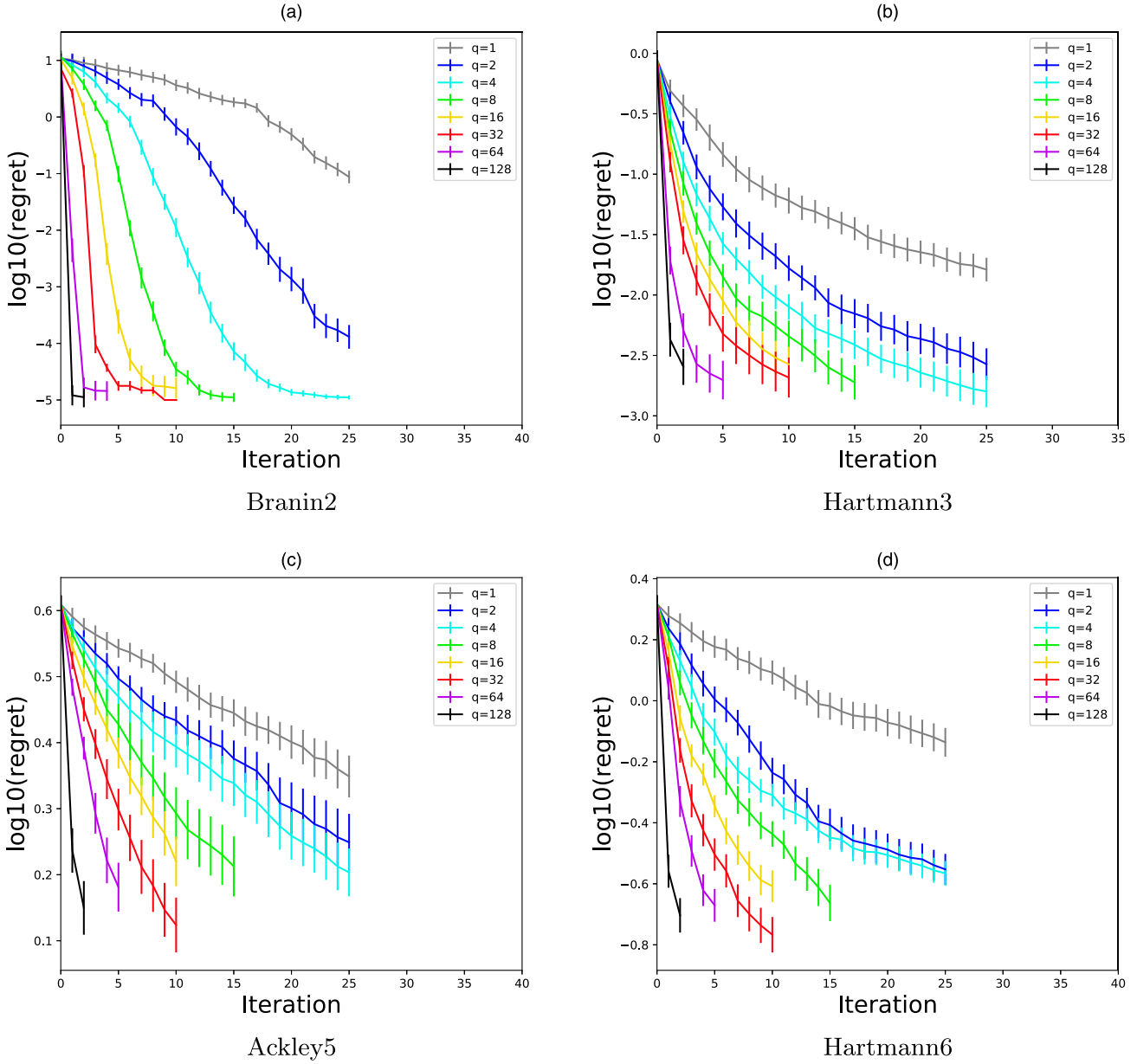


Notes. $\log_{10}$(regret) versus iteration for MOE-qEI (solid line) and CL-mix (dashed line), where the error bars show 95% confidence intervals obtained from 100 repeated experiments with different sets of initial points. MOE-qEI converges faster with better solution quality than the heuristic method CL-mix for all q.

This formula requires q calls of the q-dimensional multivariate normal CDF ($\Phi_q(\cdot,\cdot)$), and q^2 calls of the $q-1$ dimensional multivariate normal CDF ($\Phi_{q-1}(\cdot,\cdot)$). Because computing multivariate normal CDFs, which are often implemented with numerical integration or Monte Carlo sampling (Genz 1992), is expensive to evaluate even for moderate q, calculating this analytically formula quickly becomes slow and numerically challenging as q grows.

Although Chevalier and Ginsbourger (2013) did not propose using this closed-form formula to solve the inner optimization problem (5), one can adapt it to this purpose by using it within any derivative-free optimization method. We implemented this approach in the MOE package, where we use the L-BFGS (Liu and Nocedal 1989) solver from SciPy (Jones et al. 2001) as the derivative free optimization solver. We call this approach "Benchmark 1."

Figure 2. Comparison Against EGO

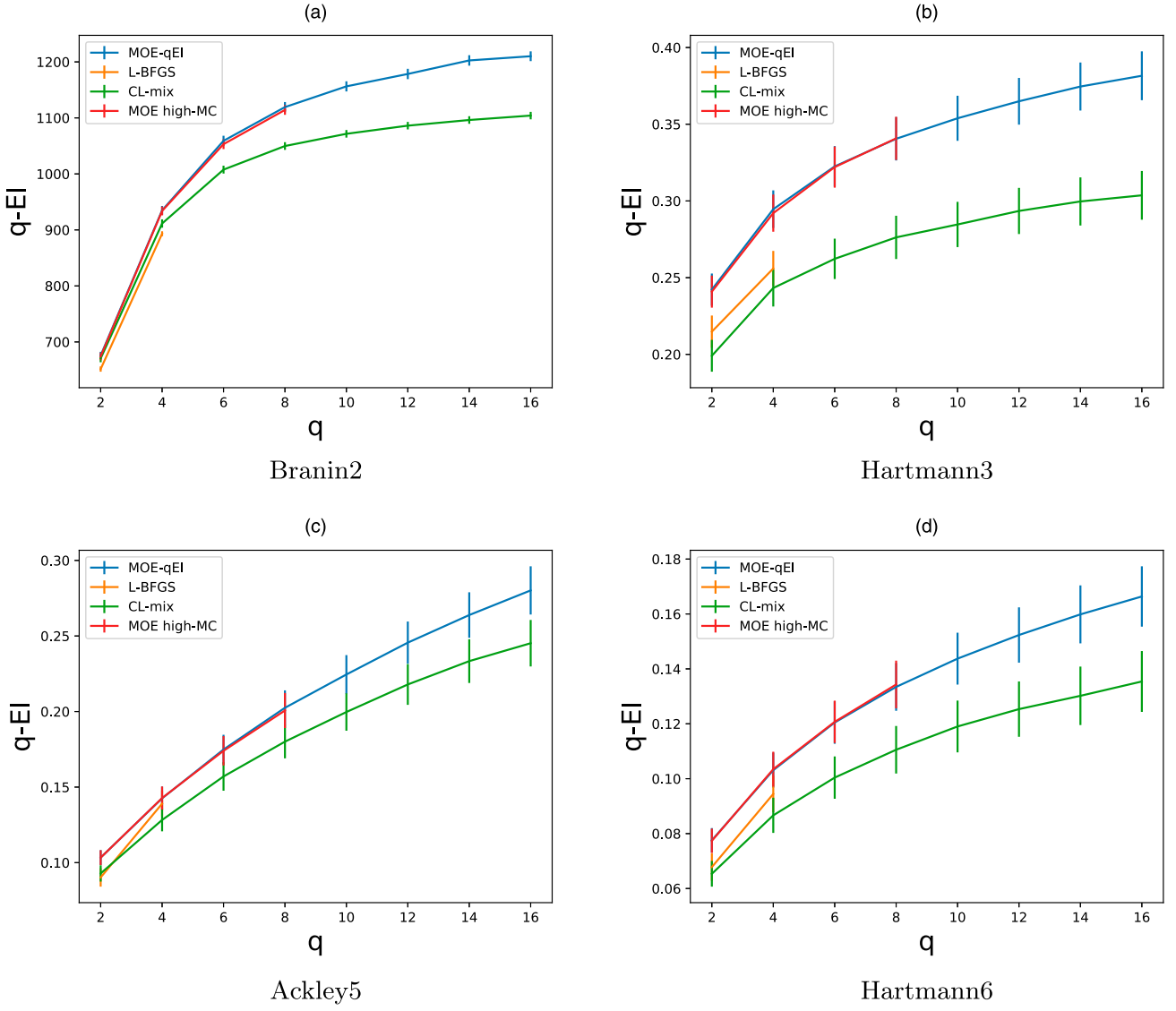


Note. $\log_{10}$(regret) versus iteration for different q, where the error bars show 95% confidence intervals obtained from 100 repeated experiments with different sets of initial points.

We compare MOE-qEI, CL-mix, and Benchmark 1 in solving the inner optimization problem, in terms of both solution quality and runtime, as shown in Figures 3 and 4. Without surprise, MOE-qEI achieves the best solution quality among the three, and its running time is almost comparable with CL-mix, which is expected to be the fastest approach because it sacrifices solution quality for speed. We ran Benchmark 1 with q going only up to 4 because its runtime goes up drastically with q. MOE-qEI's runtime scales as well as q grows, making it feasible to run in applications with high parallelism. To our surprise, CL-mix achieves competitive solution quality against Benchmark 1, using only a fraction of Benchmark 1's runtime. Therefore, despite the promise of using the closed-form formula for *q-EI* to fully solve the inner optimization problem, this formula's long runtime and the slow convergence of L-BFGS due to lack of derivative information make Benchmark 1 a less favorable option than CL-mix in practice.

Figure 3 also includes a method called "MOE high-MC." This method runs MOE-qEI on GPU with the number of Monte Carlo samples M for the gradient estimator set to 10^7, much higher than the default

Figure 3. Comparison of Algorithms for Solving the Inner Optimization Problem



Notes. Solution quality (maximum *q-EI*) versus q for different algorithms solving the inner optimization problem. For each test function, we generated 500 instances of the inner optimization problem by randomly sampling $(2d + 2)$ points, and the plot shows the average solution quality and 95% confidence interval for the expected solution quality over 500 problem instances.

setting of 1,000. As shown in the figure, the solution quality for MOE high-MC is the same as that of MOE-qEI, which confirms that $M = 1000$ is sufficiently large.

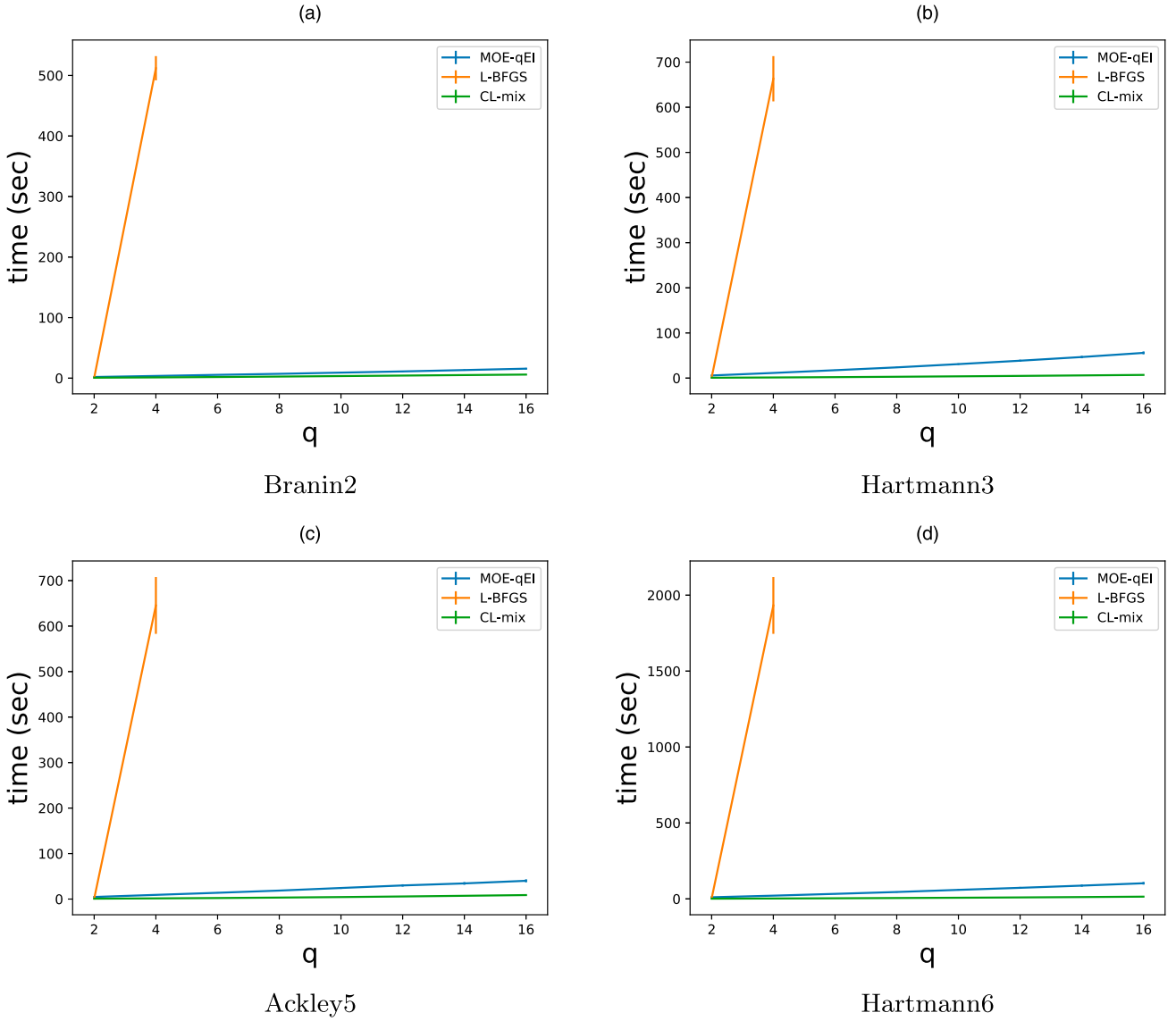
5.3. Comparison on Evaluation of ∇q-*EI*

A recently published book chapter, Marmin et al. (2015), developed independently and in parallel to this work, proposed a method for computing ∇q-*EI* using a closed-form formula derived from (17) and then proposed to use this formula inside a gradient-based optimization routine to solve (5). The formula is complex and therefore we do not reproduce it here.

This formula faces even more severe computational challenges than (17); indeed, it requires $O(q^4)$ calls to multivariate normal CDFs with dimension between $(q - 3)$ and q. Because computing high-dimensional multivariate normal CDFs is itself challenging, this closed-form evaluation becomes extremely time-consuming.

MOE-qEI's Monte Carlo based approach to evaluating ∇q-*EI* offers three advantages over using the closed-form formula: first, numerical experiments suggest that computation scales better with q; second, it can be easily parallelized, with significant speedups possible through parallel computing on GPUs, as is implemented within the MOE library; third, by using a small number of replications to make each iteration run quickly, and by using it within a stochastic gradient ascent algorithm that averages noisy gradient

Figure 4. Comparison of Algorithms for Solving the Inner Optimization Problem



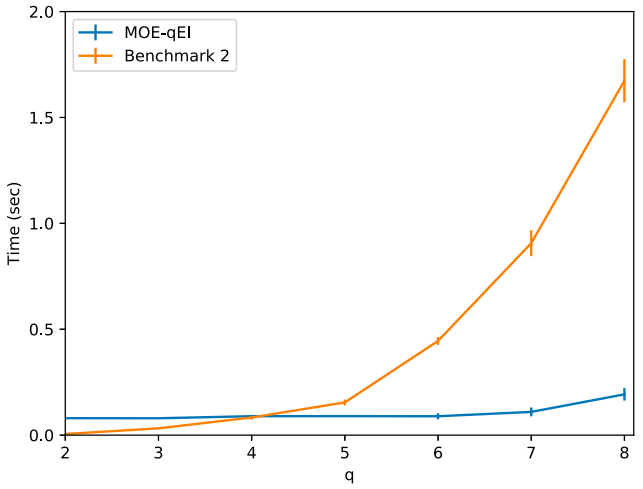
Note. Runtime versus q for different algorithms solving the inner optimization problem.

information intelligently across iterations, we may more intelligently allocate effort across iterations, only spending substantial effort to estimate gradients accurately late in the process of finding a local maximum.

We first show that computation of exact gradients using our gradient estimator with many replications on a GPU scales better with q through numerical experiments. We compare with closed-form gradient evaluation on a CPU as implemented in the "DiceOptim" package (Ginsbourger et al. 2015) and call it "Benchmark 2." We computed $\nabla q\text{-}EI$ at 200 randomly chosen points from a two-dimensional design space to obtain a 95% confidence interval for the average computation time. To make the gradient evaluation in MOE-qEI close to exact, we increased the number of Monte Carlo samples used in the gradient estimator to 10^7, which ensures that the variance of each component of the gradient is on the order of 10^{-10} or smaller for all q we consider in our experiments. Given the large number of Monte Carlo samples, we use the GPU option in the MOE package to speed up computation. This GPU implementation is made possible by the trivial parallelism supported by our Monte Carlo based gradient estimator, whereas a massively parallel GPU implementation of closed-form gradient evaluation would be more challenging.

Figure 5 shows that computational time for Benchmark 2 increases quickly as q grows, but increases slowly for MOE-qEI's Monte Carlo estimator, with this Monte Carlo estimator being faster when $q \geq 4$. This difference in performance arises because gradient

Figure 5. Comparison Against Closed-Form Evaluation of $\nabla q\text{-}EI$



Notes. Average time to compute $\nabla q\text{-}EI$ with high precision versus q, comparing the gradient-based estimator from MOE-qEI using a large number of samples (10^7) in a parallel GPU implementation with the closed-form formula from Marmin et al. (2015). The stochastic gradient estimator in MOE-qEI scales better in q and is faster when $q \geq 4$.

estimation in MOE-qEI focuses Monte Carlo effort on calculating a single high-dimensional integral, whereas the closed-form formula decomposes this high-dimensional integral of interest into a collection of other high-dimensional integrals that are almost equally difficult to compute, and the size of this collection grows with q.

We may reduce the number of Monte Carlo samples used by MOE-qEI substantially while still providing high-accuracy estimates: if we reduce M from 10^7 to 10^4, the variance of each component of the gradient remains below 10^{-7}. Because the GPU implementation provides a roughly 100× to 1,000× speedup, the CPU-only implementation of our stochastic gradient estimator with this reduced value of M has run-time comparable with or better than the GPU-based results pictured in Figure 5, showing that even without the hardware advantage offered by a GPU, our stochastic gradient estimator provides high-accuracy estimates faster than Marmin et al. (2015) for $q \geq 4$.

Moreover, because stochastic gradient ascent is tolerant to noisy gradients, we may obtain additional speed improvements by reducing the number of Monte Carlo samples even further. Using fewer Monte Carlo samples in each iteration has the potential to increase efficiency by only putting effort toward estimating the gradient precisely when we are close to the stationary point, which stochastic gradient ascent performs automatically through its decreasing stepsize sequence. Thus, our stochastic gradient estimator is both faster when using a large number of samples to produce essentially exact estimates, and it offers more flexibility in its ability to produce inexact estimates at low computational cost.

Although our numerical experiments have focused on synchronous evaluations, it is also of interest to understand the value of our approach for asynchronous evaluations (Section 3.4). In the synchronous case, we saw that the value provided by our approach relative to other methods tended to grow with q. This occurs, first, because other methods for computing or approximating the q-EI and its gradient do not scale as well with q; and, second, because the benefit to optimization provided by derivatives grows with the dimension of the problem solved, which is qd for synchronous evaluations.

In the asynchronous setting, the difficulty of computing or approximating q-EI continues to be determined by q, but the dimension of the inner optimization problem is instead determined by p, which is often small. Thus, although we do expect our approach will outperform alternatives in the asynchronous setting, especially for large q, we expect that methods that use derivative-free optimization (Chevalier and Ginsbourger 2013) would be more competitive in the asynchronous case than in the synchronous one. Nevertheless, we expect that using derivatives, such as the ones offered by our approach, would typically outperform derivative-free optimization even for small p.

6. Conclusions

We proposed an efficient method based on stochastic approximation for implementing a conceptual parallel Bayesian global optimization algorithm proposed by Ginsbourger et al. (2008). To accomplish this, we used infinitesimal perturbation analysis (IPA) to construct a stochastic gradient estimator and showed that this estimator is unbiased. We also provided convergence analysis of the stochastic gradient ascent algorithm with the constructed gradient estimator. Through numerical experiments, we demonstrate that our method outperforms the existing state-of-the-art approximation methods.

References

Ababou R, Bagtzoglou AC, Wood EF (1994) On the condition number of covariance matrices in kriging, estimation, and simulation of random fields. *Math. Geology* 26(1):99–133.

Amatriain X (2014) 10 Lessons Learned From Building ML Systems. Recording of presentation from MLconf 2014. Accessed November 26, 2015, https://www.youtube.com/watch?v=WdzWPuazLA8.

Boender CGE, Kan AR (1987) Bayesian stopping rules for multistart global optimization methods. *Math. Programming* 37(1):59–80.

Brochu E, Brochu T, de Freitas N (2010) A Bayesian interactive optimization approach to procedural animation design. *Proc. 2010 ACM SIGGRAPH/Eurographics Sympos. Comput. Animation* (Eurographics Association, Madrid), 103–112.

Calvin JM (1997) Average performance of a class of adaptive algorithms for global optimization. *Ann. Appl. Probab.* 7(3):711–730.

Calvin JM, Žilinskas A (2002) One-dimensional global optimization based on statistical models. Dzemyda G, Šaltenis V, Žilinskas A, eds. *Stochastic and Global Optimization* (Springer, New York), 49–63.

Calvin J, Žilinskas A (2005) One-dimensional global optimization for observations with noise. *Comput. Math. Appl.* 50(1):157–169.

Chevalier C, Ginsbourger D (2013) Fast computation of the multi-points expected improvement with applications in batch selection. *Internat. Conf. Learning Intelligent Optim.* (Springer, New York), 59–69.

Clark S (2014) Introducing "MOE": metric optimization engine; a new open source, machine learning service for optimal experiment design. Accessed November 26, 2015, http://engineeringblog.yelp.com/2014/07/introducing-moe-metric-optimization-engine-a-new-open-source-machine-learning-service-for-optimal-ex.html.

Clark SC, Liu E, Frazier PI, Wang J, Oktay D, Vesdapunt N (2014) Metrics optimization engine. Accessed September 17, 2017, http://yelp.github.io/MOE/.

Dennis JE Jr, Torczon V (1991) Direct search methods on parallel machines. *SIAM J. Optim.* 1(4):448–474.

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

Frazier PI, Wang J (2016) Bayesian optimization for materials design. Lookman T, Alexander FJ, Rajan K, eds. *Information Science for Materials Discovery and Design* (Springer, New York), 45–75.

Frazier PI, Xie J, Chick SE (2011) Value of information methods for pairwise sampling with correlations. *Proc. Winter Simulation Conf.* (IEEE, Washington DC), 3979–3991.

Genz A (1992) Numerical computation of multivariate normal probabilities. *J. Comput. Graphics Statist.* 1(2):141–149.

Ginsbourger D (2009) Two advances in Gaussian process-based prediction and optimization for computer experiments. Report, MASCOT09 Meeting, MASCOT09, Villetaneuse, France.

Ginsbourger D, Le Riche R, Carraro L (2008) A multi-points criterion for deterministic parallel global optimization based on Gaussian processes. Working Paper hal-00260579, Ecole Nationale Superieure des Mines, Sainte-Etienne, France.

Ginsbourger D, Le Riche R, Carraro L (2010) Kriging is well-suited to parallelize optimization. Tenne Y, Goh CK, eds. *Computational Intelligence in Expensive Optimization Problems* (Springer, New York), 131–162.

Ginsbourger D, Picheny V, Roustant O, Binois M, Chevalier C, Marmin S, Wagner T (2015) Diceoptim: Kriging-based optimization for computer experiments. Accessed February 13, 2016, https://cran.r-project.org/web/packages/DiceOptim/index.html.

Ho YC (1987) Performance evaluation and perturbation analysis of discrete event dynamic systems. *IEEE Trans. Automat. Control.* 32(7):563–572.

Holland JH (1992) *Adaptation in Natural and Artificial Systems: an Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence* (MIT Press, Cambridge, MA).

Howard RA (1966) Information value theory. *IEEE Trans. Systems Sci. Cybernetics* 2(1):22–26.

Huang D, Allen TT, Notz WI, Zeng N (2006) Global optimization of stochastic black-box systems via sequential kriging meta-models. *J. Global Optim.* 34(3):441–466.

Jamil M, Yang XS (2013) A literature survey of benchmark functions for global optimisation problems. *Internat. J. Math. Model. Numer. Optim.* 4(2):150–194.

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. *J. Global Optim.* 13(4):455–492.

Kennedy J (2010) Particle Swarm Optimization. Sammut C, Webb GI, eds. *Encyclopedia of Machine Learning* (Springer, New York), 760–766.

Kim SH, Nelson BL (2007) Recent advances in ranking and selection. *Proc. 39th Conf. Winter Simulation* (IEEE Press, Washington, DC), 162–172.

Kushner H, Yin GG (2003) *Stochastic Approximation and Recursive Algorithms and Applications*, vol. 35 (Springer Science & Business Media, Berlin).

Kushner HJ (1964) A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *J. Fluids Engrg.* 86(1):97–106.

Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. *Math. Programming* 45(1–3):503–528.

Marmin S, Chevalier C, Ginsbourger D (2015) Differentiating the multipoint expected improvement for optimal batch design. Pardalos P, Pavone M, Farinella GM, Cutello V, eds. *Machine Learning, Optimization, and Big Data* (Springer, New York), 37–48.

McKay MD, Beckman RJ, Conover WJ (2000) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 42(1):55–61.

Mockus J (1989) *The Bayesian Approach to Local Optimization* (Springer, New York).

Mockus J, Tiesis V, Zilinskas A (1978) The application of Bayesian methods for seeking the extremum. Dixon LCW, Szego GP, eds. *Toward Global Optimization*, vol. 2 (North-Holland, Amsterdam), 117–129.

Polyak BT (1990) New stochastic approximation type procedures. *Automat. i Telemekh* 7(2):98–107.

Rasmussen C, Williams C (2006) *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, MA). Accessed January 29, 2020, http://www.gaussianprocess.org/gpml.

Ruppert D (1988) Efficient estimations from a slowly convergent Robbins-Monro process. Technical Report, Cornell University Operations Research and Industrial Engineering, Ithaca, NY.

Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989) Design and analysis of computer experiments. *Statist. Sci.* 4(4):409–423.

Smith SP (1995) Differentiation of the Cholesky algorithm. *J. Comput. Graph. Statist.* 4(2):134–147.

Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. *Proc. 25th Internat. Conf. Neural Inform. Processing Systems*, vol. 2 (Curran Associates Inc., Red Hook, NY), 2951–2959.

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

Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, et al. (2019) SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. Preprint submitted July, https://ui.adsabs.harvard.edu/abs/2019arXiv190710121V.

Villemonteix J, Vazquez E, Walter E (2009) An informational approach to the global optimization of expensive-to-evaluate functions. *J. Global Optim.* 44(4):509–534.

Xie J, Frazier PI, Chick SE (2016) Bayesian optimization via simulation with pairwise sampling and correlated prior beliefs. *Oper. Res.* 64(2):542–559.

Jialei Wang is co-founder and chief scientist of SensesAI, an artificial intelligence software company that builds next-generation banking systems powered by AI. His research is focused on automated machine learning, and Bayesian optimization. He holds a PhD in operations research from Cornell University.

Scott C. Clark is co-founder and CEO of SigOpt, Inc., an optimization software company that helps firms worldwide tune complex models. Scott holds a PhD in applied mathematics from Cornell University.

Eric Liu is a technical lead in the applied machine learning group at Yelp. He works on research and development for various machine-learning applications and is currently focused on revenue management: acquisition, retention, forecasting, etc. Eric holds an MS in aeronautical engineering from MIT.

Peter I. Frazier is an associate professor in the School of Operations Research and Information Engineering at Cornell University and a staff data scientist at Uber. His research interests lie in Bayesian optimization, multi-armed bandit, and incentive design for social learning. He is the winner of the Air Force Office of Scientific Research Young Investigator Award, the National Science Foundation CAREER Award, and best paper awards from various venues.