Triangulation candidates for Bayesian optimization

Robert B. Gramacy
Department of Statistics
Virginia Tech
Blacksburg, VA 24060, USA
rbg@vt.edu

Annie Sauer
Department of Statistics
Virginia Tech
Blacksburg, VA 24060, USA
anniees@vt.edu

Nathan Wycoff
McCourt School of Public Policy
Georgetown University
Washington DC, 20057, USA
nathan.wycoff@georgetown.edu

Abstract

Bayesian optimization involves "inner optimization" over a new-data acquisition criterion which is non-convex/highly multi-modal, may be non-differentiable, or may otherwise thwart local numerical optimizers. In such cases it is common to replace continuous search with a discrete one over random candidates. Here we propose using candidates based on a Delaunay triangulation of the existing input design. We detail the construction of these "tricands" and demonstrate empirically how they outperform *both* numerically optimized acquisitions and random candidate-based alternatives, and are well-suited for hybrid schemes, on benchmark synthetic and real simulation experiments.

1 Introduction

We address the continuous unconstrained optimization problem

$$x^\star = \operatorname*{argmin}_{x \in \mathcal{B}} f(x) \tag{1}$$

where the bounding box $\mathcal{B}$ is a hyperrectangle, often taken as $[0,1]^d$ in coded inputs. The objective $f : \mathcal{B} \to \mathbb{R}$ is a *blackbox* function, meaning that we can only learn about its behavior through expensive, often simulation-based, evaluation. Such problems are most challenging when f is highly non-convex, and thus contains multiple local minima. A tacit goal of a solver is to minimize the number of times that f is evaluated in search of a global solution.

The earliest papers on Bayesian optimization (BO) adapted statistical modeling and design principles to tackle this optimization problem [Močkus, 1975, Jones et al., 1998]. Applications on physics-based simulators f are provided by Pourmohamad and Lee [2021]; scenarios in machine learning are reviewed by Garnett [2022]. BO is common in studies of engagement and user experiences in online platforms [e.g., Letham and Bakshy, 2019], hyperparameter estimation for deep learning [e.g. Turner et al., 2021, Feurer et al., 2018] and materials design [e.g. Zhang et al., 2020b], to name a few.

To illustrate BO and introduce our contributions, consider $f(x) = \sin(x)$ and $\mathcal{B} = [-1, 2\pi + 1]$ evaluated at six equally-spaced inputs x. Next fit a surrogate $\hat{f}_n$, to data (X_n, Y_n), where $y_i = f(x_i)$, for $i = 1, \ldots, n = 6$. We privilege Gaussian process (GP) based-surrogates [e.g., Gramacy, 2020],

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

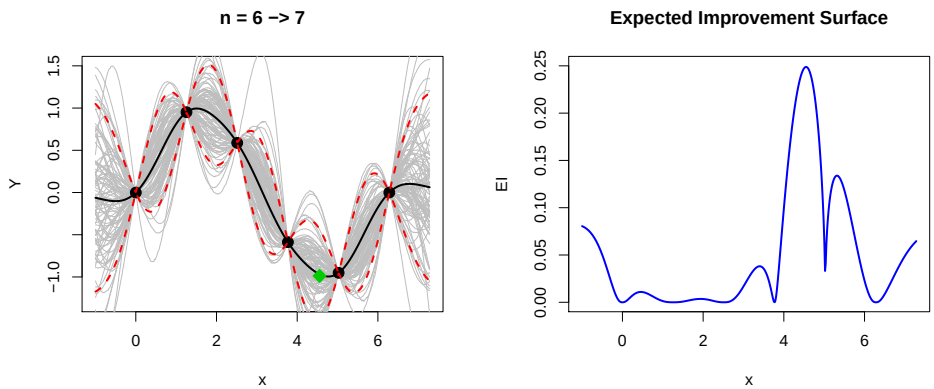

Figure 1: Predictive surface (left) via means (solid black), 90% intervals (dashed-red), and sample paths (gray); EI (right) and resulting acquisition (green diamond).

but our methodology is agnostic to that choice so long as the predictive equations from $\hat{f}_n$ have similar features – e.g., non-linear predictive mean $\mu_n(x)$ and higher predictive uncertainty/standard deviation $\sigma_n(x)$ away from the training data sites.

The left panel of Figure 1 shows a fitted $\hat{f}_6$ via μ_6 as solid black line with error bars $\mu_6 \pm 2\sigma_6$ in dashed-red. Each of the one-hundred gray lines is a sample from the predictive distribution. This distribution interpolates the training data (black dots), a GP hallmark, yet our contributions are not limited to noise free settings. (We entertain a noisy f in Section 3.2.) The current best run $f_n^{\min} = \min\{y_1, \ldots, y_n\}$ is near $x = 5$, but there is considerable predictive uncertainty about other parts of the input space $\mathcal{B}$, relative to $f_n^{\min}$. Locations in $[-1, 0] \cup [4, 6] \cup [2\pi, 2\pi + 1]$ are promising.

BO *acquisition criteria* serve to operationalize this notion. We shall focus on two and note that several others are variations on similar themes. One is *Thompson sampling* [TS; Thompson, 1933], and involves drawing from the predictive distribution $Y(x) \sim \hat{f}_n \mid X_n, Y_n$ and choosing $x_{n+1} = \operatorname{argmin}_{x \in \mathcal{B}} Y(x)$. TS amounts to randomly selecting one of those gray lines and optimizing it in lieu of the expensive f, so the criterion is inherently stochastic. The other is *expected improvement* [EI; Jones et al., 1998]. Define *improvement* as $I(x) = \min\{0, f_n^{\min} - Y(x)\}$ and take its expectation with respect to $Y(x)$. If $Y(x) \sim \hat{f}_n$ is Gaussian, as with GPs, then this has closed form:

$$\mathrm{EI}(x) = \mathbb{E}\{I(X)\} = \int I(x)\, dY(x) = (f_n^{\min} - \mu_n(x))\, \Phi(z_n(x)) + \sigma_n(x)\, \phi(z_n(x)), \qquad (2)$$

where $z_n(x) = (f_n^{\min} - \mu_n(x))/\sigma_n(x)$ and Φ/ϕ are the Gaussian cdf/pdf. For non-Gaussian $\hat{f}_n$, one can always resort to Monte Carlo (MC) integration instead. The right panel of Figure 1 provides EI for $\hat{f}_6$. Contrary to TS, the EI acquisition $x_{n+1} = \operatorname{argmax}_{x \in \mathcal{B}} \mathrm{EI}(x)$ can be deterministic, if properly solved. The maximum EI acquisition x_7 is shown as a green diamond on the left panel.

Observe that TS and EI involve "inner-optimizations" over a criterion, a task that may be even more challenging than the original problem (1). Each gray line is highly multi-modal, as is the EI surface. Both have about as many local optima as there are training data points n, whereas f has only two local minima in $\mathcal{B}$. Speedy evaluation of $\mu_n(x)$ and $\sigma_n(x)$ relative to $f(x)$ saves us, but only partly. We still desire good x_{n+1} without exhaustive search or cumbersome subroutines.

Two strategies are common, sometimes separately, sometimes in tandem as a hybrid. The simplest option is to distribute N candidate points $\mathcal{X}_N$ throughout the input space, evaluate the criterion on $\mathcal{X}_N$, and thereby replace a continuous search with a discrete one. In low input dimension a dense grid of candidates can be used effectively. In higher dimension one can populate $\mathcal{X}_N$ with a space-filling design like a random Latin hypercube sample [LHS; Mckay et al., 1979] to manage the computational expense of evaluating the criterion exhaustively. A higher-powered approach is to locally apply a smooth, convex optimization library such as L-BFGS-B [Byrd et al., 2003]. Derivatives may be approximated by finite-differencing or autograd [Paszke et al., 2017], or may have a simple closed form depending on the criterion and nature of $\hat{f}_n$. The former requires more computer work and some consideration of numerical stability; the latter more researcher/programmer effort when possible. (MC-based $\hat{f}_n$ or EI challenges both approaches.) Pure candidate-based search is most common with TS because of its stochastic nature. Gradient-based continuous search is popular in simple GP/EI

setups, but a multi-start scheme is essential to avoid inferior local solutions. This is where the hybrids come in: candidates seeding local solvers.

In this paper we contend that both strategies, random candidates and multi-start local optimization, can be replaced by (or hybridized with) more thoughtfully chosen $\mathcal{X}_N$. In the 1d setting of Figure 1 we could place candidates at the midway points between each of the existing n inputs and the boundary $\mathcal{B}$, implementing a kind of multi-pronged bisection search [Burden and Faires, 1985, Section 2.1] and resulting in $N = n + 1 = 7$ candidates. The best of those $\mathcal{X}_N$ by either criterion may not give a precise solution to the inner optimization, but it would be an effective one because EI and many of the random gray lines indicate a solution close to one of those midway points. Such locations might be much better than ones identified by a limited candidate or numerical local search.

This 1d example is overly simplistic. Going forward, we shall explicitly target two and higher dimensions. In Section 2 we scale-up the midway candidate idea, also suggested by Scott et al. [2011], to what we call "tricands", based on Delaunay triangulation and the convex hull of X_n. We explore tricands' features and limitations and suggest remedies with the BO application in mind. In Section 3 we demonstrate that tricands outperform both random LHS candidates and a multi-start gradient-based numerical inner optimization in the conventional setting where surrogate GP predictive equations are available in closed form. In Section 4 we consider two nonstationary surrogates requiring Markov chain Monte Carlo (MCMC), for which closed-form acquisition criteria are not readily available. Candidates are essential in this setting, and our tricands are better than random space-filling ones. Our discussion in Section 5 emphasizes tricands' "plug-n-play" nature – they can be inserted into any candidate-based scheme – and suggests potential for extension.

2 Delaunay triangulation candidates

Many criteria for BO resemble Eq. (2), balancing exploitation ($\mu_n(x)$ below $f_n^{\min}$) with exploration (large $\sigma_n(x)$). Most surrogates inflate predictive uncertainty ($\sigma_n(x)$) away from training data locations X_n. In the case of GPs, this is what produces the "sausage shaped" predictive intervals shown as red-dashed lines in Figure 1. Our main insight is that careful allocation of candidates *between* existing training data locations, where $\sigma_n(x)$ is high, allows for BO acquisitions that do not necessitate cumbersome numerical optimization of posterior predictive quantities but still balance exploitation and exploration. A hard statistical optimization can be replaced with an easier geometric one.

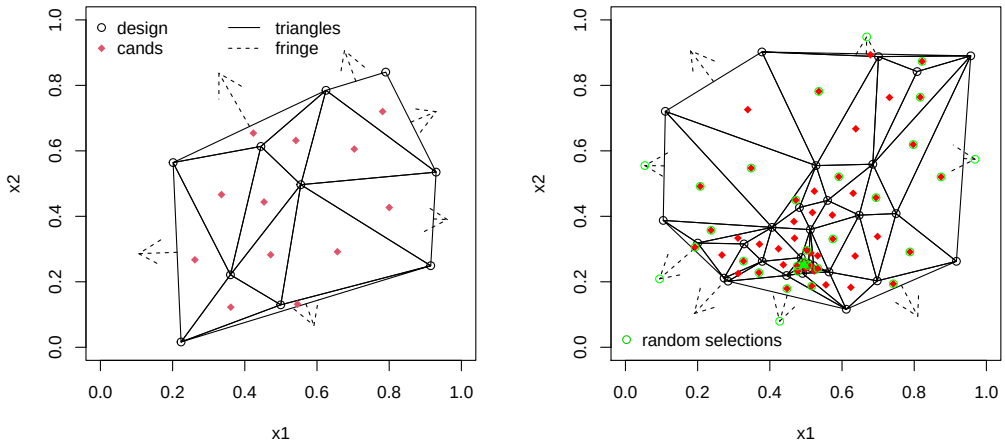

Figure 2: Interior and fringe candidates (both), and randomly sub-sampled candidates (right).

The idea is sketched graphically in Figure 2, whose details will emerge over the next three subsections. The existing design X_n is shown as open circles, and the selected candidates are closed red dots *and* the tips of dashed arrows. We call these interior (Section 2.1) and fringe (Section 2.2) candidates, respectively. Both are calculated based on Delaunay triangles and their convex hull, outlined by solid black lines in the figure, and explained next. We provide this illustration in 2d to ease visualization; however our Section 3–4 benchmarks include higher dimensional input spaces.

2.1 Interior candidates

A *Delaunay triangulation* of X_n is an angle-maximizing set of d line-segments connecting geographically nearby points, such that no point lies inside the circumcircle of those points. Such triangulations subdivide the interior of the *convex hull* of X_n, which is the the smallest (convex) set that contains all points. In 2d, lines depicting those subdivisions form triangles. In higher dimension they form tetrahedra, however one often abuses the nomenclature and still refers to triangles. For more details, see, e.g., Lee and Schachter [1980]. The solid lines of Figure 2 indicate a triangulation for random X_n (left) and an X_n derived after BO iterations (right), described in Section 3 .

There are fast algorithms for calculating Delaunay triangulations. In 2d, these have $\mathcal{O}(n \log n)$ runtimes. Higher dimensional analysis is complicated by the number of triangles, which depends on the geometry of X_n, a topic we shall return to shortly. For R we use the `geometry` package on CRAN [Habel et al., 2019]; for Python we use `Delaunay` in `scipy.spatial` [Virtanen et al., 2020]. Both are wrappers around the C library `Qhull`, implementing "quickhull" [Barber et al., 1996]. Bates and Pronzato [2001] first suggested Delaunay triangulation for BO. That early work only explored two input dimensions, possibly because they did not have convenient access to `Qhull`. They also did not entertain the BO-specific extensions we provide here, particularly in Sections 2.2–2.3.

Let the triangles be denoted by T_j, for $j = 1, \ldots, n_T$. Each T_j is a $(d+1) \times d$ matrix when X_n has d columns. Create n_T new candidates $\mathcal{X}_{n_T}$ where the jth candidate is located at the *barycenter* of T_j:

$$\tilde{x}_j = \bar{T}_j = \frac{1}{d+1} \sum_{i=1}^{d+1} T_j[i,] \quad \text{or} \quad \tilde{x}_{jk} = \frac{1}{d+1} \sum_{i=1}^{d+1} T_j[i,k], \quad \text{for } k = 1, \ldots, d.$$

The left expression is vectorized over the second, column dimension of T_j. The second is explicit about coordinates $\tilde{x}_j^\top = (\tilde{x}_{j1}, \ldots, \tilde{x}_{jd})$. Red dots in Figure 2 provide a visual. These locations will almost certainly not be the maximal points of $\sigma_n(x)$ in the vicinity of T_j, but they will be close because $\tilde{x}_j$ is within T_j but far from its edges. We refer to these $\mathcal{X}_{n_T}$ as "interior" candidates.

In 2d, Euler's formula gives that $n_T = 2n - 2 - h(X_n)$ where $h(X_n)$ is the number of elements of X_n on its convex hull. In Figure 2, $n = 10$ and $h(X_n) = 6$ so $n_T = 12$. When $d \geq 3$, the number of faces of the tetrahedra can grow as $n^{\lceil d/2 \rceil}$ depending on the nature of X_n. Figure 3 provides

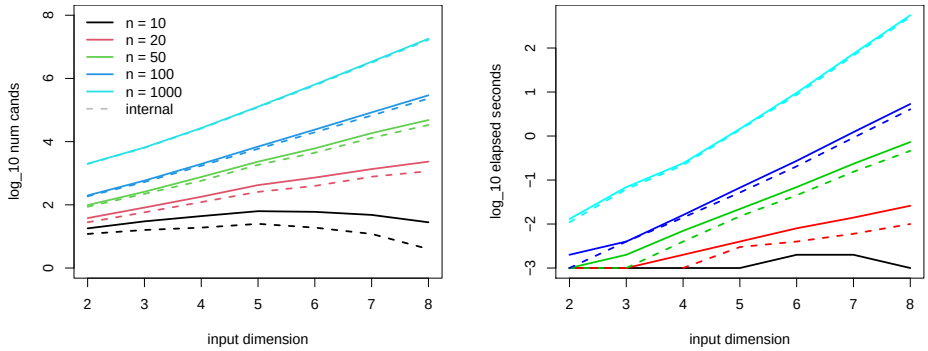

Figure 3: Tricands N (left) over input dimension d and design size n, and compute time (right) using a single core of an Intel i7-6900K CPU at 3.20GHz.

an empirical view of the number of candidates (left) and triangulation compute time (right), for varying d and n where X_n is distributed uniformly at random in $\mathcal{B} = [0,1]^d$. The dashed lines in the left panel are $\log_{10} n_T$. Notice that when n is small, n_T decreases as d increases. For fixed d, n_T steadily increases with n. In our BO context this is a good thing, because so too does the modality of acquisition criterion. With nonparametric surrogates like GPs, the complexity of the response surface can increase with training data size n. The search effort for the next acquisition should be commensurate with this complexity, an innate characteristic of (interior) tricands. The right panel of Figure 2, described later in Section 2.3, shows how this works after many iterations of EI acquisition.

2.2 Fringe candidates

Having candidates $\mathcal{X}_{n_T}$ only in the interior of the convex hull could limit exploration if the complement of the volume of the hull and $\mathcal{B}$ is large. It could limit exploitation if the solution is on/near the boundary of $\mathcal{B}$. One remedy is to force X_n to contain boundary points, such as the corners of the input space. This is not a bad approach in low dimension (e.g., in 2d there are only four corners), but could be prohibitive in higher dimension. Later in Section 3 we consider a $d = 8$ example, having 256 corners in $\mathcal{B} = [0, 1]^8$, which would nearly blow our entire budget of runs.

We instead prefer the "fringe" candidates pointed to by the dashed arrows in Figure 2. There is one of these for each facet (edge in 2d) of the convex hull, extending perpendicularly from middle of the facet half way to the boundary of $\mathcal{B} = [0, 1]^d$. The `Qhull` library furnishes both ingredients: $d \times d$ facets F_j and normal d-vectors $\vec{v}_j$ for $j = 1, \ldots, n_F$. Using these quantities, let $\bar{F}_j = \frac{1}{d}\sum_{i=1}^{d} F_j[j,]$ denote the coordinates of the middle of facet j. Then, the jth fringe candidate in $\mathcal{X}_{n_F}$ is

$$\vec{x}_j = \bar{F}_j + \tfrac{1}{2}\alpha_j \vec{v}_j \quad \text{where} \quad \alpha_j = \min\left\{\mathbb{I}_{\{\vec{v}_j > 0\}} - \bar{F}_j \vec{v}_j\right\}.$$

Above, "min" is picking out the nearest boundary to $\bar{F}_j$. Division by two mirrors the bisection search analogy for interior candidates, but this could be a tuning parameter. Non-unit rectangular $\mathcal{B}$ is doable, but requires a more convoluted formula. More general $\mathcal{B}$ may present challenges.

Fringe and interior candidates may be combined: $\mathcal{X}_N = [\mathcal{X}_{n_T}; \mathcal{X}_{n_F}]$, stacked row-wise to form an $N \times d$ matrix with $N = n_T + n_F$. The number of fringe candidates n_F is generally small compared to n_T. This is shown empirically in the left panel of Figure 3, where N is indicated by the solid line, and n_T by the dashed one. As n increases, the gap between N and n_T, indicating n_F, narrows.

2.3 Targeted sub-sampling

Having N grow exponentially in n may not suit all applications. Entertaining $N \approx 10{,}000$ candidates when $n = 100$ and $d = 6$, referring to Figure 3, is cumbersome and potentially overkill. One way to limit N is random sub-sampling: simply calculate the full $\mathcal{X}_N$ based on X_n and downsample, uniformly at random, a subset of size $N_{\text{sub}} < N$. Since our triangulation strategy was designed to focus on exploration (finding locally high $\sigma_n(x)$) via midway candidates, we find it advantageous to guarantee retaining some of those candidates which are promising for exploitation (low $\mu_n(x)$), still without any explicit numerical optimization – only geometry.

One of the rows of X_n corresponds to $f_n^{\min}$, the best input found so far: $x_n^{\min} = x_i$ s.t. $i = \text{argmin}_{i=1,\ldots,n} y_i$ (or $\mu_n(x_i)$ in the noisy case). Let $\mathcal{T}_n^{\min} = \{T_j : x_n^{\min} \in T_j, j = 1, \ldots, n_T\}$ denote the set of triangles containing $x_n^{\min}$, and similarly let $\mathcal{X}_n^{\min} = \{\tilde{x}_j \in \mathcal{X}_N : T_j \in \mathcal{T}_n^{\min}\}$ denote the candidates associated with those triangles. Those are, in a geometric sense, adjacent to $x_n^{\min}$. A fringe candidate may also be considered adjacent in an analogous way, however we place them in the complement $\mathcal{X}_N \setminus \mathcal{X}_n^{\min}$. Now, rather than sub-sample uniformly from the full set $\mathcal{X}_N$, we partition sampling from points adjacent to $x_n^{\min}$, i.e., from $\mathcal{X}_n^{\min}$, and from points farther afield in $\mathcal{X}_N \setminus \mathcal{X}_n^{\min}$. In so doing, we guarantee that our N_{sub} candidates cover potential for exploitation and exploration, respectively. We prefer a 10:90 split, with up to 10% of N_{sub} coming from candidates adjacent to $x_n^{\min}$, fewer if $|\mathcal{X}_n^{\min}| < N_{\text{sub}}$, and likewise 90% from its complement.

The right panel of Figure 2 provides an illustration after $n = 30$ runs optimizing the Goldstein–Price function with EI under random initialization (details in Section 3.1). When $n = 30$ we have $N \approx 60$, with the precise value depending on $h(X_n)$. Here we consider $N_{\text{sub}} = 30$. Observe how randomly sub-sampled candidates $\mathcal{X}_{N_{\text{sub}}}$, circling the original red candidates $\mathcal{X}_N$ in green, concentrate near the global minimum $(0.5, 0.25)$ because $x_n^{\min}$ is in the vicinity. Other sub-sampled candidates are spread out more widely. This figure illustrates how acquisitions, and thus candidates, gravitate toward promising regions for exploitation without neglecting areas of potential exploration. A ridge of local minima may be found in the banana-shaped region traced out by a concentration of X_n and $\mathcal{X}_N$.

2.4 Implementation and software

Our implementation, provided for Python and R in our git repository,[1] is relatively tidy. For example, `tricands.R` therein contains just 71 lines of code, eleven of which are to support optional

[1] `http://bitbucket.org/gramacylab/tricands`

visualizations in 2d such as those in Figure 2. The heavy lifting is done by `Qhull`. When performing Delaunay triangulations, we provide option `"Q12"` to work around numerical instabilities that sometimes arise. When calculating convex hulls, option `"n"` returns normal vectors used in calculating fringe candidates (Section 2.2).

Defaults yield both fringe and interior candidates and fix a maximum candidate size of $\texttt{max} = N_{\text{sub}} = 100d$, but these are user-adjustable. In experiments coming shortly, we deliberately limit N_{sub} even further so that a fairer comparison can be made to other continuous search and candidate-based methods. When $N \leq N_{\text{sub}}$, all interior and fringe candidates are returned. The user may supplement these with additional $N_{\text{sub}} - N$ random candidates if desired. If $N_{\text{sub}} < N$, the execution flow looks for a variable `best`, providing the index of $x_n^{\min}$ in X_n, making sure about 10% of tricands come from adjacent triangles. When `best=NULL`, tricands are sub-sampled at random.

3 Classical GP benchmarking

In this first of two sections on benchmarking, we focus on BO via traditional GP surrogates. We follow a homework problem in Section 7.4 of Gramacy [2020], which piggy-backs off of GP, EI, and TS demonstrations earlier in the chapter. Software and other particulars are relegated to Appendix A.1. All of our examples are fully reproducible using the code provided in our git repository. We consider three methods for solving EI acquisitions: a continuous search of the criterion via L-BFGS-B with 5-multi-starts, LHS candidates, and tricands. For TS acquisitions, we similarly employ both LHS candidates and tricands. As non-BO (not surrogate-based) comparators, serving primarily as benchmarks, we entertain "raw" L-BFGS-B and Nelder–Mead [Nelder and Mead, 1965].

Two examples are showcased here, with a third example relegated to Appendix A.2. Our synthetic fs, including those in Section 4, are described in more detail on the pages of the Virtual Library for Simulation Experiments [VLSE; Surjanovic and Bingham, 2013]. In all of our experiments we code inputs to $[0, 1]^d$ and summarize results for 100 random restarts where each surrogate is initialized with the same (unique to each random restart) starting design of size $n_0 = 12$, except in Section 3.2 where we use $n_0 = 60$. This design is taken uniformly at random following the advice of Zhang et al. [2021] who caution that small space-filling initial designs can spark pathological behavior in BO. We track "best observed value" (BOV) $f_n^{\min}$ as a measure of progress which is summarized by median over $n = 1, \ldots, n_{\text{end}}$ and by boxplots for particular n along the way.

3.1 Goldstein–Price

The Goldstein–Price function is a popular low-dimensional (2d) benchmark for BO [e.g., Picheny et al., 2012]. A total of $n_{\text{end}} = 50$ acquisitions are entertained, and all candidate-based methods (tricands and LHS) are limited to fifty candidates. This means $\texttt{max} = N_{\text{sub}} = 50$ for tricands, with fewer candidates when $N < N_{\text{sub}}$. The top row of Figure 4 summarizes results in three views. In the top-left panel, median progress in BOV is shown over increasing budgets of evaluations (n) as if each subsequent acquisition were the last. In the top-middle and right panels, boxplots capture the distribution of BOV at $n = 30$ and $n = 50$, respectively. In the top-left panel, methods based on a multi-start numerical local search use solid lines; those based on tricands are dashed; those based on LHS candidates are dotted. (Nelder–Mead is an exception, being red-dashed.) In the boxplots, tricands use slightly heavier ink so that they stand out. Text printed at the top of the top-right panel indicates the average number of times the criterion, EI or TS, was evaluated in solving the inner optimization sub-problem(s), cumulative over all acquisitions. In the case of a numerically optimized EI (labeled "EI"), these happen within iterations of L-BFGS-B search.

Beginning with medians over n in the top-left panel, observe that the dashed lines (those based on tricands) are uniformly superior to all other comparators. EI with tricands is slightly better than TS with tricands. Only at the very end of the run does the raw Nelder–Mead alternative look competitive. At the 30th evaluation (top-middle) the boxplots indicate that there are some MC repetitions where BOV based on tricands are underperforming. However, more than half of the time these are the best two (i.e., both EI and TS with tricands) of all. TS with tricands is the winner in terms of best-case performance, whereas the EI version has slightly better worst-case behavior. Neither local optimizer (red) is competitive. Finally, in the top-right panel we see that the story is similar, except that now EI with tricands edges out its TS analog. Although median performance for Nelder–Mead is competitive, more than half of the time its BOV at $n_{\text{end}} = 50$ is among the two worst in the experiment.

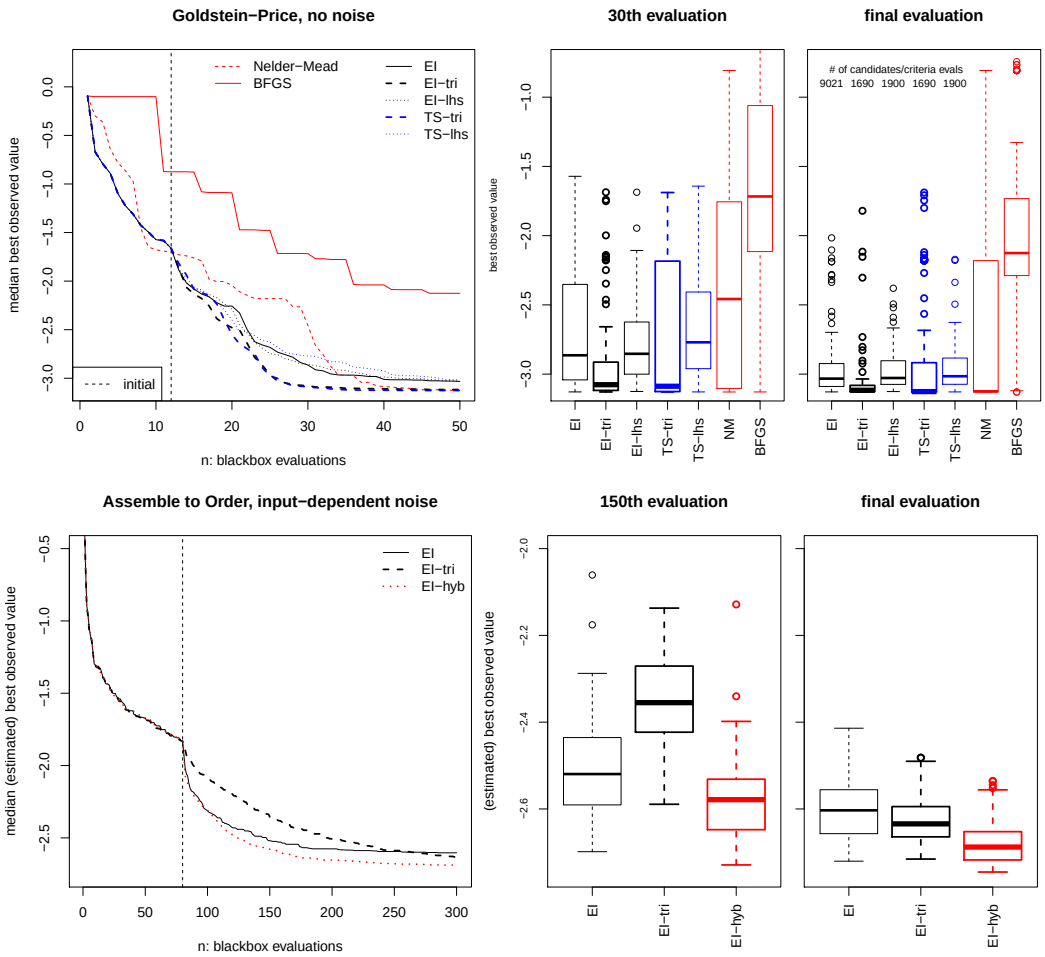

Figure 4: Goldstein–Price (top) and ATO (bottom) BOV over 100 MC trials: median (left panel) and distribution at intermediate (middle) and final (right) acquisitions. Suffixes "-tri" or "-lhs" denote a candidate search. Absence of a suffix (e.g., "EI" alone) indicates multi-start L-BFGS-B.

Perhaps the most striking result from the figure is that tricands-based EI ("EI-tri") outperforms its multi-start numerical analog ("EI") despite a factor of five fewer evaluations of the EI criterion (about 9000 compared to 1700). A more aggressive, gradient-based search misutilizes computational resources. Finding a precise solution – ultimately furnishing a maximal local value in an inferior domain of attraction – may come at the expense of finding an accurate approximation near a global solution. The same is true, but to a lesser extent, when comparing TS variations (5% reduction).

See Appendix A.2 for nearly identical results with the higher dimensional Hartmann 6 function.

3.2 Assemble to order (ATO)

The assemble-to-order (ATO) simulator [Hong and Nelson, 2006] deploys a queuing system virtualizing a manufacturer receiving orders for a variety of widgets, each with different demand for component parts. Target inventory levels for eight parts comprise the controllable inputs ($d = 8$). The output is profit, a scalar distilling inventory and order fulfillment costs under random orders for parts, fulfillment and inventory replenishment times. ATO is coded in Matlab and exhibits input-dependent noise. Although it is reasonably fast (seconds per evaluation), relatively large n is required to separate signal from noise ($n_0 = 80$ and $n_{\text{end}} = 300$). Cubic computational bottlnecks for GP inference are in play here, despite using a thrifty heteroskedastic GP [Binois et al., 2018]. More details, including software, are provided in Appendix A.1. Since our Matlab licenses prevented us from running parallel instances, we had to limit our experiment somewhat to obtain results in a reasonable amount

of time. Therefore we only entertained three acquisition alternatives: random multi-start numerically optimized EI, tricands with the default $N_{\text{sub}} = 100d = 800$, and a hybrid using the best tricands candidate to initialize a local numerically optimized EI ("EI-hyb").

The bottom panels of Figure 4 track BOV for these three alternatives in a view similar to the panels above. Note that the original ATO problem involves maximization, so we have negated the output and also scaled it so that its units were more similar to our other benchmark problems. Since the output is random, the BOVs are estimates. We used the surrogate fit at n_{end} to assign in-sample predictions via $\mu_{n_{\text{end}}}(x_i)$. Otherwise, the story is similar to our earlier synthetic results. Tricands' progress is initially slower than numerically optimized EI, but eventually gives lower BOV values. The difference at n_{end} is slight, but statistically significant. A paired Wilcox test with alternative hypothesis that BOV for "EI-tri" is below "EI" has a p-value of 0.0037. The hybrid is even better, beating pure tricands and pure numerical optimization 85 and 88 out of 100 times, respectively.

4 Sampling-based surrogates

Tricands are most valuable when the inner-optimization problem cannot be solved by library-based numerical methods, even locally. This happens when the surrogate predictive surface is discontinuous and/or when inference requires MCMC. Our examples here involve response surfaces exhibiting nonstationarity, meaning that the input–output dynamics evolve over the input space. This demands a more elaborate surrogate. We consider two. A treed Gaussian process [TGP; Gramacy and Lee, 2008] uses axis-aligned partitioning with GPs. Such divide-and-conquer excels when dynamics change abruptly across individual inputs creating distinct regimes. MCMC can average over the location of probable partition boundaries. We use the R package `tgp` on CRAN [Gramacy, 2007], following Section 4 of Gramacy and Taddy [2010] for candidate-based EI acquisition.

Deep Gaussian processes [DGPs; Damianou and Lawrence, 2013] warp inputs to accommodate a more subtle evolution of dynamics in the input space compared to the abrupt regime changes of TGP. Although variational inference is popular for DGPs [Salimbeni and Deisenroth, 2017], Sauer et al. [2020] argue that in active learning contexts, such as BO, full posterior integration leads to better uncertainty quantification, and thus better acquisitions. Here we use the R package `deepgp` on CRAN [Sauer, 2021] which supports EI evaluation on candidates.

Unfortunately, neither `tgp` nor `deepgp` facilitate TS acquisition for BO. However, the `tgp` package furnishes a maximum *a posteriori* sample which can be used as the basis of a local search [Gramacy and Taddy, 2010, Section 4]. Here we show that tricands provides better candidates for both pure candidate EI search and this hybrid scheme.

4.1 Abrupt changes

The Gramacy & Lee (G&L) function benefits from a model supporting hard breaks even though its dynamics evolve smoothly. The 2d input domain (coded to $[0, 1]^2$) is mostly flat except in one quadrant where a local maximum is twinned with a (global) minimum. The domain of attraction of that global minimum covers only about 10% of the input space. It is easily missed by random initial designs and candidate sets. TGP is able to isolate the interesting quadrant with just two axis aligned partitions, thus recognizing that sampling effort should be concentrated there.

The top row of Figure 5 shows results in the same three views as earlier. The number of candidates is limited to twenty, otherwise the setup is unchanged. First ignore the red lines and boxplots (those labeled "hyb-") and focus on the black (TGP) and green (DGP). Notice that in both cases the tricands-based comparators, dashed and/or bolded, outperform their LHS-based analogues. In terms of medians over n (top-left panel), that dominance is uniform. In terms of boxplots, the disparity is stark with the exception of a few outlying DGP-based BOV values with tricands. TGP seems to edge out DGP (with tricands) which we attribute to the abrupt change between the interesting quadrant and the rest of the input space. Now focus on the red "hyb-" results. These are the ones where TGP candidate-based search is finished with a gradient-based EI search on the maximum *a posteriori* model. Again, tricands wins. To supplement the visuals, we report that the tricands hybrid BOV value was lowest in 97/100 reps (top-right).

A higher dimensional example using the Michaelwicz function is provided in Appendix A.3.

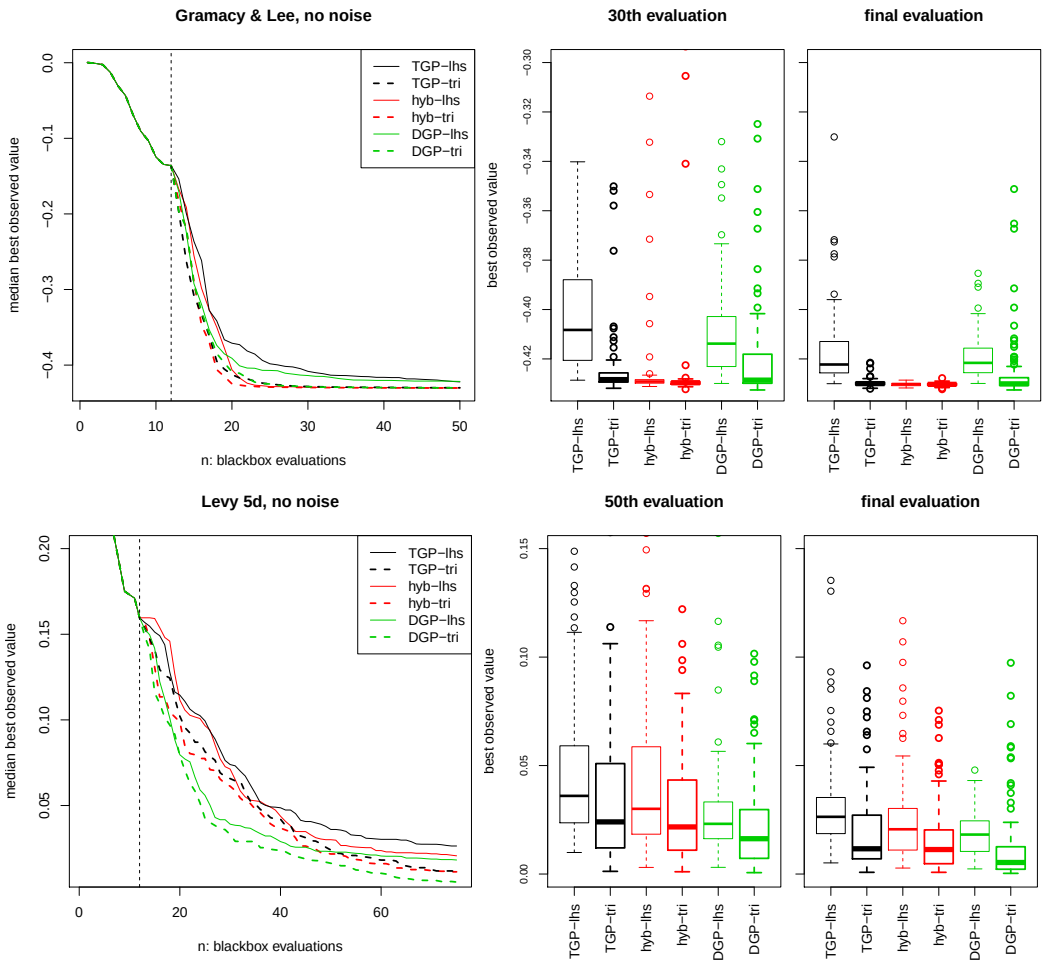

Figure 5: Gramacy & Lee (top) and 5d Levy (bottom) with MCMC-based comparators.

4.2 Smooth changes

The Levy function is defined in arbitrary dimension. It is ruffled with many peaks and valleys. Here we consider $d = 5$ for variety; see the bottom row of Figure 5. Commensurate with our other examples, we consider two hundred candidates and $n_{\text{end}} = 75$.

Both candidate DGP variations outperform their TGP counterparts. Slowly/smoothly varying dynamics favor warping inputs as opposed to hard partitioning. This is evident in both median and full distribution (boxplot) views. In the median view (bottom-left), notice that the dashed lines are uniformly under the solid ones of the same color. Tricands are providing better median performance over all n. Although the preeminence of tricands is apparent visually, we report that it provided better BOV in 81/100 MC restarts among the best (DGP) comparators. When tricands are involved, the hybrid candidate/numerical option is not discernibly better than the pure candidate alternative.

5 Discussion

We offer a novel take on space-filling candidates for acquisition in BO. The idea is to fill the spaces in-between previous acquisitions. This is motivated by an analogy to bisection search, but also by the nature of GP predictive surfaces which often serve as surrogates in BO. GP surrogates have organically inflated uncertainty between training data sites, which makes those spots attractive for BO acquisition. This notion is extended to the space between the convex hull of existing training data and the boundary $\mathcal{B}$ of the study region. In an array of benchmark exercises we have demonstrated that these tricands lead to superior performance in BO compared to both higher-powered gradient-based

acquisition schemes and simpler space-filling candidates. Tricands' main attraction is that they mimic the behavior of higher-powered searches with the implementation simplicity of candidates. That simplicity means that tricands may be deployed where the higher-powered alternatives cannot, such as when the surrogate is not continuous or requires MCMC.

Additional discussion on high input dimension and other extensions is provided in Appendix B.

Acknowledgments and Disclosure of Funding

We are grateful for funding from DOE LAB 17-1697 via subaward from Argonne National Laboratory for SciDAC/DOE Office of Science ASCR and High Energy Physics, and for comments from two teams of anonymous referees which greatly improved the paper.

References

Dario Azzimonti, David Ginsbourger, Clément Chevalier, Julien Bect, and Yann Richet. Adaptive design of experiments for conservative estimation of excursion sets. *Technometrics*, pages 1–14, 2020.

Evan Baker, Pierre Barbillon, Arindam Fadikar, Robert B Gramacy, Radu Herbei, David Higdon, Jiangeng Huang, Leah R Johnson, Pulong Ma, Anirban Mondal, et al. Analyzing stochastic computer models: A review with opportunities. *arXiv preprint arXiv:2002.01321*, 2020.

C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, December 1996. ISSN 0098-3500. doi: 10.1145/235815.235821. URL `https://doi.org/10.1145/235815.235821`.

R. Bates and L. Pronzato. Emulator-based global optimisation using lattices and delaunay tesselation. In *Sensitivity Analysis of Model Output*, pages 189–192, Madrid (Espagne), 2001.

Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22(3):773–793, 2012.

Henrik Bengtsson. *R.matlab: Read and Write MAT Files and Call MATLAB from Within R*, 2018. URL `https://CRAN.R-project.org/package=R.matlab`. R package version 3.6.2.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

M Binois, RB Gramacy, and M Ludkovski. Practical heteroscedastic Gaussian process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics*, 27(4):808–821, 2018. doi: 10.1080/10618600.2018.1458625. URL `https://doi.org/10.1080/10618600.2018.1458625`.

M Binois, J Huang, RB Gramacy, and M Ludkovski. Replication or exploration? Sequential design for stochastic simulation experiments. *Technometrics*, 27(4):808–821, 2019. doi: 10.1080/00401706.2018.1469433. URL `https://doi.org/10.1080/00401706.2018.1469433`.

Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to bayesian optimization, 2021.

Mickaël Binois and Robert B. Gramacy. hetgp: Heteroskedastic Gaussian process modeling and sequential design in `r`. *Journal of Statistical Software*, 98(13):1–44, 2021. doi: 10.18637/jss.v098.i13. URL `https://www.jstatsoft.org/index.php/jss/article/view/v098i13`.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

AD Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct):2879–2904, 2011.

R.L. Burden and J.D. Faires. *Numerical Analysis*. PWS Publishers, 1985.

Richard Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16, 02 2003. doi: 10.1137/0916069.

Rob Carnell. `lhs`*: Latin Hypercube Samples*, 2018. URL `https://CRAN.R-project.org/package=lhs`. R package version 0.16.

Clément Chevalier and David Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In *International Conference on Learning and Intelligent Optimization*, pages 59–69. Springer, 2013.

Clément Chevalier, Julien Bect, David Ginsbourger, Emmanuel Vazquez, Victor Picheny, and Yann Richet. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. *Technometrics*, 56(4):455–465, 2014.

D Austin Cole, Robert B Gramacy, James E Warner, Geoffrey F Bomarito, Patrick E Leser, and William P Leser. Entropy-based adaptive design for contour finding and estimating reliability. *arXiv preprint arXiv:2105.11357*, 2021.

Andreas Damianou and Neil D Lawrence. Deep Gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective Bayesian optimization over high-dimensional search spaces, 2021.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 5497–5508, 2019.

Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. Practical transfer learning for bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2018.

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

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A multi-points criterion for deterministic parallel global optimization based on kriging. In *NCP07*, 2007.

J Gonzalez, M Osborne, and N Lawrence. `GLASSES`: Relieving the myopia of Bayesian optimisation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 790–799, 2016.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial intelligence and statistics*, pages 648–657. PMLR, 2016.

RB Gramacy. `tgp`: an R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models. *Journal of Statistical Software*, 19(9):6, 2007.

RB Gramacy. `laGP`: large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software*, 72(1):1–46, 2016.

RB Gramacy and Herbert KH Lee. Optimization under unknown constraints. In *Bayesian Statistics*, volume 9. Oxford University Press, 2011.

RB Gramacy and HKH Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

RB Gramacy and F Sun. `laGP`*: Local Approximate Gaussian Process Regression*, 2018. URL `http://bobby.gramacy.com/r_packages/laGP`. R package version 1.5-3.

RB Gramacy and MA Taddy. Categorical inputs, sensitivity analysis, optimization and importance tempering with `tgp` version 2, an R package for treed Gaussian process models. *Journal of Statistical Software*, 33(6): 1–48, 2010.

Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida, 2020. `http://bobby.gramacy.com/surrogates/`.

Kai Habel, Raoul Grasman, Robert B. Gramacy, Pavlo Mozharovskyi, and David C. Sterratt. `geometry`*: Mesh Generation and Surface Tessellation*, 2019. URL `https://CRAN.R-project.org/package=geometry`. R package version 0.4.5.

L.J. Hong and B.L. Nelson. Discrete optimization via simulation using compass. *Operations Research*, 54(1): 115–129, 2006.

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

Remi Lam, Karen Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Advances in Neural Information Processing Systems*, 29:883–891, 2016.

ER Leatherman, TJ Santner, and AM Dean. Computer experiment designs for accurate prediction. *Statistics and Computing*, pages 1–13, 2017.

Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

Benjamin Letham and Eytan Bakshy. Bayesian optimization for policy search via online-offline experimentation. *J. Mach. Learn. Res.*, 20:145–1, 2019.

Alexandre Marques, Remi Lam, and Karen Willcox. Contour location via entropy reduction leveraging multiple information sources. In *Advances in Neural Information Processing Systems*, pages 5217–5227, 2018.

D Mckay, Richard Beckman, and William Conover. A comparison of three methods for selecting vales of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 05 1979.

Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

JA Nelder and R Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NeurIPS Autodiff Workshop*, 2017.

V Picheny, T Wagner, and D Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, pages 1–20, 2012.

Tony Pourmohamad and Herbert KH Lee. *Bayesian Optimization with Application to Computer Experiments*. Springer, New York, NY, 2021.

Pritam Ranjan, Derek Bingham, and George Michailidis. Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, 50(4):527–541, 2008.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *arXiv preprint arXiv:1705.08933*, 2017.

Annie Sauer. `deepgp`*: Sequential Design for Deep Gaussian Processes using MCMC*, 2021. URL `https://CRAN.R-project.org/package=deepgp`. R package version 0.3.0.

Annie Sauer, Robert B Gramacy, and David Higdon. Active learning for deep Gaussian process surrogates. *arXiv preprint arXiv:2012.08015*, 2020.

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

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Peter Su and Robert L Scot Drysdale. A comparison of sequential delaunay triangulation algorithms. *Computational Geometry*, 7(5-6):361–385, 1997.

S Surjanovic and D Bingham. Virtual library of simulation experiments: test functions and datasets. `http://www.sfu.ca/~ssurjano`, 2013.

MA Taddy, HKH Lee, GA Gray, and JD Griffin. Bayesian guided pattern search for robust local optimization. *Technometrics*, 51(4):389–401, 2009.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*, 2021.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *Advances in Neural Information Processing Systems*, 33:19511–19522, 2020.

Boya Zhang, Robert B Gramacy, Leah Johnson, Kenneth A Rose, and Eric Smith. Batch-sequential design and heteroskedastic surrogate modeling for delta smelt conservation. *arXiv preprint arXiv:2010.06515*, 2020a.

Boya Zhang, D Austin Cole, and Robert B Gramacy. Distance-distributed design for Gaussian process surrogates. *Technometrics*, 63(1):40–52, 2021.

Yichi Zhang, Daniel W Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1):1–13, 2020b.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Attempts were made throughout to justify early claims.
   (b) Did you describe the limitations of your work? [Yes] Some enumerated in Section 2; extended discussion in Appendix B.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We are aware of none.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Our contributions are not theoretical.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] A URL is provided in the text. Everything is fully reproducible by sourcing the scripts therein.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] A high level description is given in Sections 3–4, further detail in Appendix A.1, and comments are provided within our scripts in the git repo.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Boxplots in our main results figures provide a summary of those distributions.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Figure 2, right panel provides timings for our main tricands calculations.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] All existing libraries are cited as well as the methodological papers that describe the original ideas.
   (b) Did you mention the license of the assets? [No] These are standard libraries on prominent repositories, and licensing information can be found on those pages.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] A link to a git repository is provided.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Implementation and additional empirical results

Here we summarize implementation details and experimental results that were removed from the main body of the paper due to space constraints. All of the empirical work in our paper is fully reproducible. Code may be found in our git repository: `http://bitbucket.org/gramacylab/tricands`.

A.1 Classical GP implementation details

The GP surrogate for the Goldstein–Price and Hartman 6 examples (Section 3 and Appendix A.2, respectively) used the `laGP` package for R [Gramacy, 2016, Gramacy and Sun, 2018] on CRAN with separable/automatic-relevance determination lengthscales in a Gaussian kernel using gradient-based MLE. We supplied our own EI function, following the prototype provided in Chapter 7 of Gramacy [2020], modified to keep track of the number of evaluations. Finite-differencing was used to approximate gradients. For Latin hypercube samples we used the `lhs` [Carnell, 2018] package for R. "Raw" comparators L-BFGS-B and Nelder–Mead [Nelder and Mead, 1965] local optimizers were facilitated via the `optim` function in R.

The heteroskedastic GP surrogate used for ATO (Section 3.2) was via `hetGP` [Binois and Gramacy, 2021] for R. The built-in EI capability in that package, which supports both numerical optimization (via analytic gradients) and candidates, unfortunately could not easily be modified to keep track of the number of evaluations. Simulations, which require Matlab, were run in R through the `R.matlab` interface [Bengtsson, 2018]. The input space for ATO is discrete, in $\{1, \ldots, 20\}^8$, which is different from the real-valued inputs of our other examples. The only modification we made to accommodate this nuance was to "snap" acquisitions to that grid, implemented in coded units. Sometimes this resulted in replications in the design. No additional consideration was given to replicates, despite findings that they are beneficial in similar contexts [Binois et al., 2019].

Details for the surrogates used in Section 4 are provided therein.

A.2 Hartmann 6

As a second example of conventional GP-based BO, consider the six-dimensional Hartmann function. Again, see Picheny et al. [2012] for more on this benchmark. Our experimental setup is identical the Goldstein–Price experiment in Section 3 except here we use the default $N_{\text{sub}} = 100d = 600$. Figure 6 summarizes results. Even with this high N_{sub}, tricands still result in many fewer acquisition criteria evaluations than numerically optimized EI (right panel), which must search more aggressively for the local optimum in such high dimension.

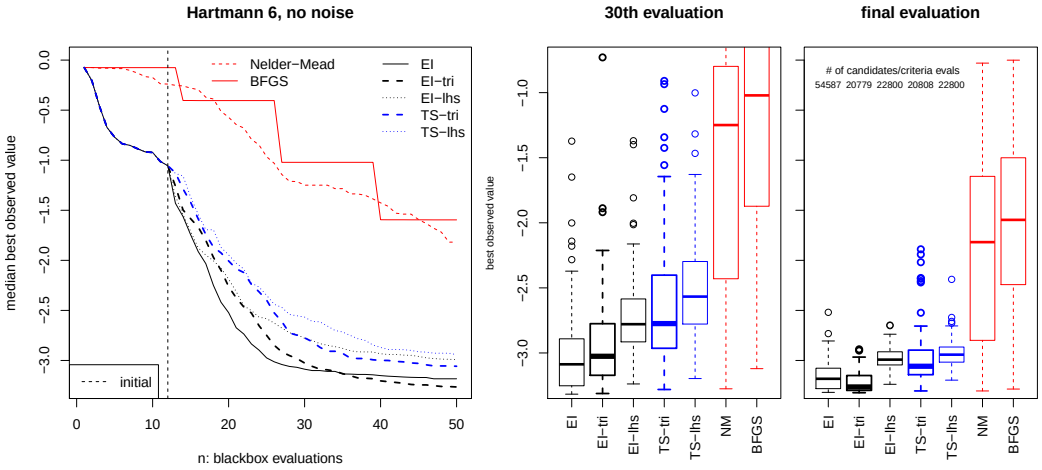

Figure 6: Analog of Figure 4 for the Hartmann 6 function.

The results are broadly similar to our earlier 2d Goldstein–Price example shown in Figure 4. Tricands-based EI yields equivalent, or better, BOV despite many fewer criteria evaluations. This is true, but to a lesser extent, with TS. It is notable that tricands came from behind in the case of EI. For the first

twenty or so acquisitions, numerically optimized EI bests its tricands analog. Geometric bias towards exploration may be less desirable in early acquisitions, but pays off by the end of the search.

A.3 Michaelwicz

As a second example of abruptly changing regimes, continuing from Section 4.1, consider the Michaelwicz function. Like G&L, the surface has large flat areas, but it also has a continuum of ridges of local minima which intersect to create deeper valleys of local minima, and ultimately one global minimum where the deepest of those ridges intersect. A nice feature of the Michaelwicz function is that it is defined in arbitrary input dimension. Here we use it in 4d, which makes for a very difficult surface to model and optimize. To cope, we take the search out to $n_{\text{end}} = 75$ and allow up to two hundred candidates per acquisition. Otherwise the setup is similar to earlier examples.

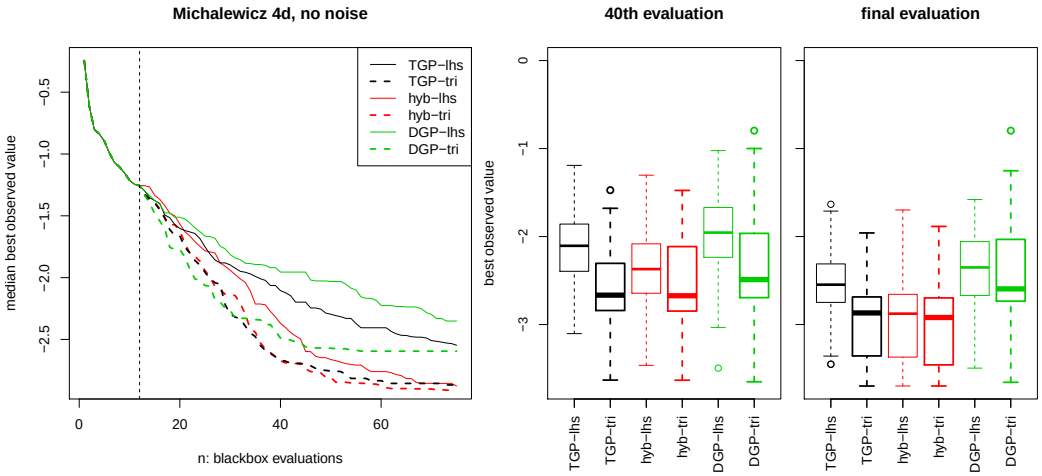

Figure 7: Michaelwicz function in 4d with MCMC-based comparators.

Figure 7 shows the results, which are largely similar to G&L, except revealing of these additional challenges. Tricands uniformly dominate LHS via median BOV. Although noise is high across all boxplots, reflecting variability in solution quality over random re-initialization, the patterns are clear in a pairwise analysis. For example, the red boxplots in the right panel look similar but, at the 75th evaluation, 75/100 tricands-based BOVs were below their LHS counterpart. As in the G&L example, TGP bests DGP. Abrupt regime changes are at odds with the DGP's smooth warping of inputs.

B Additional discussion

When reflecting on tricands' value, it is important to remember some stylized facts about BO. One is that the theory (under stringent regularity conditions) guarantees convergence to the global minimum "eventually," in the sense that with enough samples you'll explore everywhere [e.g., Bull, 2011]. That's not much help in practice, because exploring everywhere isn't practical. Another is that EI and TS are greedy; their scope is the next acquisition. Contemplating a remaining budget and entertaining optimal decisions through that lens can be caricatured as a herculean effort with marginal gains [Gonzalez et al., 2016, Frazier et al., 2008, Lam et al., 2016, Gramacy and Lee, 2011]. These are not bad ideas, but they haven't moved the needle on the *modus operandi* because they've not been incorporated into accessible libraries, and are too involved for bespoke implementation in practice.

Both theory and aggressively scoped acquisition can lose sight of the real goal. What's important in BO, and active learning in general, is creating a virtuous cycle between data acquisition and learning. Anyone who has tried knows that there is a fine line between vicious and virtuous when it comes to implementation details, despite the best of intentions and theoretical "guarantees". EI and TS are simple to implement because GP libraries are in abundance, code evaluating criteria is a few lines long, and there are many examples to cut-and-paste from. Barriers to application are low, but it's easy to get carried away to disappointment. This is what makes tricands attractive. It is motivated by simple principles: the solution is in-between your current runs, so look there. It is plug-n-play

wherever candidates are an option. This means they can be applied in situations where numerical differentiation is not available (e.g., with MC integrated surrogate). Evaluation of the acquisition criteria can be massively parallelized with candidates, say on a GPU, whereas local numerical solvers are inherently sequential.

That's not to say that tricands are a panacea. Some of our boxplots indicated that improvements over alternatives on best-observed-value (BOV) in 90/100 MC repetitions may have come at the expense of the worst case performance of the remaining 10%. This may have simply been bad luck. But if not, the deterministic nature of tricands could be to blame: not enough opportunity to be surprised. As mentioned in the main body of the paper, one can always augment tricands with random/space-filling candidates. This could be especially beneficial when N, the number of tricands for a given n, is lower than the desired budget of candidates. Entertaining tuning parameters for some of our hard-coded settings, like the distance between fringe candidates and the boundary (Section 2.2), could help. That distance could even be chosen at random. We could likewise randomly choose a location for interior candidates (Section 2.1), within each triangle, rather than taking the barycenter. When randomly downsampling (Section 2.3), we could guarantee a certain proportion of fringe candidates like we did for ones adjacent to $f_n^{\min}$.

Although the important subroutines of Delaunay triangulation and convex hulls are off-loaded to libraries, they can (at times) be computationally demanding. When n is large and d is modest, calculating N locations in the thousands (see Figure 3) could be cumbersome. Of course, the whole goal of BO is to limit n. In our experiments with $n \leq 75$, all of our triangulation/hull calculations took fractions of a second. But with big n they can take minutes (Figure 2, right panel), and that could be prohibitive. However, after each acquisition the number of new sites only increases by one ($n \to n+1$), and thus affects only a small, local part of the triangulation/hull. The `Qhull` library does not support this, but there are incremental algorithms for triangle/hull augmentation which are very fast relative to starting from scratch. For more details, see Su and Drysdale [1997].

Such a strategy might be valuable in higher dimensional BO settings. Continuous optimization theory tells us that gradient-based methods have local convergence rates independent of dimension, which has made L-BFGS-B and similar optimizers the tool of choice in solving for acquisitions. This, together with the exponential growth of input space volume, might at first blush suggest that tricands' performance is limited to low/modest dimension. However, in practice, the highly nonconvex nature of the acquisition surface significantly cheapens the theoretical results associated with gradient-based optimization. Indeed, recent work has achieved state-of-the-art performance in high d using candidate sets focused within a certain region of the input space [Eriksson et al., 2019, Wang et al., 2020, Daulton et al., 2021], though still built on traditional space-filling points such as LHS. It would be interesting to see whether replacing these space-filling points with tricands would be as beneficial in that setting as we have found it to be in modest d. Furthermore, a popular approach in scaling BO to high dimension is to reduce the input space by screening input variables or finding linear or nonlinear embedding spaces, rendering the problem a low dimensional one (see Binois and Wycoff [2021] for an overview). The aim of such an approach is in part to make solving the acquisition problem easier, and there's no reason to believe this wouldn't extend to tricands.

Our surrogates were GP-centric, extended to handle non-stationarity via treed partitioning and smooth (deep GP) input-warping. It would be interesting to explore the value of tricands paired with more unconventional surrogates based on trees, such as random forests [Breiman, 2001] or tree-structure Parzen estimators [Bergstra et al., 2011], where inner-optimization via gradient-based local search is a non-starter. We presented results with EI and TS-based acquisition criteria, and of course there are a litany of other heuristics. Our early experiments additionally included the upper-confidence bound [UCB; Srinivas et al., 2009] and probability of improvement (PI) criteria. The former, for most settings of the tuning parameter, mirrored our EI results whereas the latter was dominated by EI. To reduce clutter, we decided not to include them in our presentation here.

Our ATO example in Section 3.2 involved a stochastic simulator with input-dependent noise. Surrogate modeling and active learning for stochastic simulation is still very much on the frontier of the computer experiments landscape [Baker et al., 2020]. In such settings, the acquisition space should be extended to include the possibility of obtaining a replicate run, exactly duplicating one of the n existing design elements [Binois et al., 2019]. Replication can be advantageous in separating signal from noise in generic active learning tasks, and specifically in the context of BO [Binois and Gramacy, 2021, Section 4.2]. Rather than entertaining a hybrid search between a continuum of novel

locations and a discrete set of replicate sites, tricands could be leveraged to make the entire set of candidates discrete, vastly simplifying the inner optimization search.

Batch acquisition, acquiring several new runs at once, is a common paradigm in some settings. Ideas with modified EI go back at least to Ginsbourger et al. [2007] with several following thereafter [e.g., Taddy et al., 2009, Chevalier and Ginsbourger, 2013]. One, more recent approach involves penalizing regions of the input space near earlier acquisitions in the batch [González et al., 2016]. This spirit could be ported to a tricands setting: simply rule out any candidates which are in triangles T_j adjacent to those acquired earlier in the batch.

Finally, although we have emphasized BO, other active learning criteria could benefit from candidates well-spaced relative to the current design. Perhaps the most common is for active learning targeting reduced integrated mean-squared prediction error [IMSPE, e.g., Leatherman et al., 2017, Binois et al., 2019, Zhang et al., 2020a]. High input variance locations, such as between design sites, are natural candidates for reducing IMSPE. Another recently popular active learning topic in computer experiments is contour/level set finding [Ranjan et al., 2008, Bect et al., 2012, Chevalier et al., 2014, Marques et al., 2018, Azzimonti et al., 2020, Cole et al., 2021]. Many of the criteria suggested in these works involve predictive entropy from a GP surrogate, which is famously myopic; entropy (defined via a classification above and below a level set) tends to be higher near training data already near partition boundaries, leading to a clumping of acquisitions unless explicit measures are taken to spread out candidates, or to otherwise deter a numerical inner-optimizer. Tricands could offer a geometric spread of future acquisitions away from existing training data locations.