Technometrics

ISSN: 0040-1706 (Print) 1537-2723 (Online) Journal homepage: https://www.tandfonline.com/loi/utch20

Sequential Optimization in Locally Important Dimensions

Munir A. Winkel, Jonathan W. Stallrich, Curtis B. Storlie & Brian J. Reich

To cite this article: Munir A. Winkel, Jonathan W. Stallrich, Curtis B. Storlie & Brian J. Reich (2020): Sequential Optimization in Locally Important Dimensions, Technometrics, DOI: 10.1080/00401706.2020.1714738

To link to this article: https://doi.org/10.1080/00401706.2020.1714738

TECHNOMETRICS
2020, VOL. 00, NO. 0, 1–13
https://doi.org/10.1080/00401706.2020.1714738

Taylor & Francis
Taylor & Francis Group

Sequential Optimization in Locally Important Dimensions

Munir A. Winkel[a], Jonathan W. Stallrich[a], Curtis B. Storlie[b], and Brian J. Reich[a]

[a]Department of Statistics, North Carolina State University, Raleigh, NC; [b]Division of Biomedical Statistics and Informatics, Mayo Clinic, Rochester, MN

ABSTRACT

Optimizing an expensive, black-box function $f(\cdot)$ is challenging when its input space is high-dimensional. Sequential design frameworks first model $f(\cdot)$ with a surrogate function and then optimize an acquisition function to determine input settings to evaluate next. Optimization of both $f(\cdot)$ and the acquisition function benefit from effective dimension reduction. Global variable selection detects and removes input variables that do not affect $f(\cdot)$ across the input space. Further dimension reduction may be possible if we consider local variable selection around the current optimum estimate. We develop a sequential design algorithm called *sequential optimization in locally important dimensions* (SOLID) that incorporates global and local variable selection to optimize a continuous, differentiable function. SOLID performs local variable selection by comparing the surrogate's predictions in a localized region around the estimated optimum with the p alternative predictions made by removing each input variable. The search space of the acquisition function is further restricted to focus only on the variables that are deemed locally active, leading to greater emphasis on refining the surrogate model in locally active dimensions. A simulation study across multiple test functions and an application to the Sarcos robot dataset show that SOLID outperforms conventional approaches. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received April 2018
Accepted January 2020

KEYWORDS
Augmented expected improvement; Bayesian analysis; Computer experiments; Gaussian process; Local importance; Sequential design

1. Introduction

Statistical problems often involve learning about an intractable, real-valued function $f(\mathbf{x})$ that can be evaluated at given values of p continuous input variables $\mathbf{x} = (x_1, \ldots, x_p) \in [0,1]^p$. For example, physical experiments are often infeasible in many engineering problems so computer experiments are performed instead through evaluations of a computationally expensive simulator. Santner, Williams, and Notz (2003) overviewed the design and analysis of computer experiments and apply the methodology to understand the evolution of wildfires (Berk et al. 2002), to design a prosthesis device (Chang et al. 2001), and to optimize a helicopter blade design across 31 input variables (Booker et al. 1999). Jala et al. (2016) recently used computer experiments to assess the impact of electromagnetic exposure on fetuses. We focus our attention on optimization of a continuous, infinitely differentiable function, f, that is, (1) expensive to evaluate; (2) depends on a moderate to large number of variables, and (3) is measured with error.

Due to the assumed cost of evaluation, we desire an optimization strategy that requires few evaluations. For expensive f, a sequential design approach is commonly employed to find $\boldsymbol{\chi} = \arg\max_{\mathbf{x}} f(\mathbf{x})$. The approach begins with the evaluation of f at an initial design of input settings. The resulting observations are modeled with a surrogate function, $\hat{f}$, often taken to be a Gaussian process (GP) model, and $\boldsymbol{\chi}$ is estimated from $\hat{f}$. To improve this estimation, a new design point $\mathbf{x}^*$ is chosen based on an acquisition function that assigns a numeric value to each potential design point, which is related to the point's expected ability to improve estimation of $\boldsymbol{\chi}$ if it were added to the initial design. The function, f, is evaluated at $\boldsymbol{x}^*$, and $\hat{f}$ and $\hat{\boldsymbol{\chi}}$ are updated.

The sequential design process requires estimation of two optima at each step, that for $\hat{f}$ and the acquisition function. Although these functions are more tractable than f, they are still difficult to optimize in high dimensions (Kandasamy, Schneider, and Poczos 2015). Indeed, acquisition functions are often multimodal and contain regions where both the functions and their gradients are essentially zero, which is problematic for gradient-based optimization methods (Lizotte, Greiner, and Schuurmans 2012). Dimension reduction techniques are commonly employed to improve performance of maximizer estimation. Regis (2016) reduces the optimization space to a trust region centered at the current estimator $\hat{\boldsymbol{\chi}}$. Djolonga, Krause, and Cevher (2013) assumed that $f(\mathbf{x}) = g(\mathbf{A}\mathbf{x})$ for some smooth function $g(\cdot) : \mathbb{R}^q \to \mathbb{R}$ and row-orthogonal matrix $\mathbf{A} \in \mathbb{R}^{q \times p}$ with $q < p$. Their SI-BO algorithm uses low-rank approximation techniques to identify the subspace that supports f with a Bayesian bandit framework for optimization with respect to g. Wang et al. (2016) proposed the REMBO algorithm which uses a similar dimension reduction technique but identifies maximizers within randomly generated embeddings $\mathbf{z} = \mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is randomly generated.

A special case of the SI-BO and REMBO algorithms could require $\mathbf{A}\mathbf{x}$ to simply produce a selection of $q < p$ input variables, that is, to have the algorithms remove variables from consideration. For example, the "importance" of each variable may be quantified through a sensitivity analysis that assesses the variability of f as $\mathbf{x}$ changes over each dimension (Shan

CONTACT Jonathan W. Stallrich jwstalli@ncsu.edu Department of Statistics, North Carolina State University, 2311 Stinson Dr., 27695-8203 Raleigh, NC 27695-8203.

Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/TECH.

and Wang 2010). If that variability is reasonably large (small), then the variable is called globally active (inactive). To this end, Linkletter et al. (2006) specified mixture priors on the GP parameters and use Markov chain Monte Carlo (MCMC) to determine the posterior probabilities of each variable being globally active. The globally inactive variables are removed from the design and analysis, and the resulting lower-dimensional space is easier to optimize across.

Even after employing global variable selection, further dimension reduction is possible if we were to focus our attention on a localized region of the input space, similar to the idea of a trust region. For example, local sensitivity examines the partial derivatives of f evaluated at a particular input $\mathbf{x}^*$ (Oakley and O'Hagan 2004), say at $\hat{\boldsymbol{\chi}}$. Bai et al. (2014) proposed two approaches for local variable selection. The first assumes a local linear model around some input $\mathbf{x}$, assesses variable importance using local sensitivity, and implements a penalized LASSO framework to perform local variable selection. The second approach uses a forward/backward stepwise approach to choose the set of locally active variables around $\mathbf{x}$ using local linear estimators. Zhao et al. (2018) offered a generalization of the earlier algorithms and demonstrate set convergence (of the locally active variables) as well as parameter convergence. These articles, however, do not consider using the localized information to improve estimation of $\boldsymbol{\chi}$.

While the ultimate goal is to identify $\boldsymbol{\chi} = \arg\max_{\mathbf{x}} f(\mathbf{x})$, one may not know how many additional iterations are affordable nor how many are needed to meet this goal. Therefore, we desire a sequential design approach that consistently increases $f(\hat{\boldsymbol{\chi}})$ following each sequential run. In this article, we develop a Bayesian sequential design framework called *sequential optimization in locally important dimensions* (SOLID) that accomplishes this by performing global variable selection and localized variable selection around $\hat{\boldsymbol{\chi}}$ to optimize f. In Section 2, we review Bayesian estimation of a GP, Bayesian global variable selection for response surfaces, and two common acquisition functions, expected improvement (EI) and augmented EI. We introduce in Section 3 a new measure of local variable importance, based on local changes in $\hat{f}$ near $\hat{\boldsymbol{\chi}}$ after perturbing the posterior GP parameters. In Section 4, we detail the SOLID algorithm and illustrate it on a toy example. In Section 5, we compare SOLID with standard sequential optimization methods on multiple test functions and in Section 6, demonstrate SOLID's effectiveness on a robotics dataset from Vijayakumar and Schaal (2000). We find that SOLID provides larger values of $f(\hat{\boldsymbol{\chi}})$ in the first few evaluations of f, whereas standard sequential methods require more evaluations of f to obtain comparable values of $f(\hat{\boldsymbol{\chi}})$. In Section 7, we discuss the advantages and disadvantages of using SOLID to sequentially optimize an expensive black-box function and propose some areas of further development.

2. Background

2.1. Gaussian Process Regression

Let $\mathbf{X}_0$ denote the $n_0 \times p$ initial design matrix whose rows are the p input settings of the n_0 initial runs. The success of a sequential design strongly depends on the initial design (Crombecq, Laermans, and Dhaene 2011) and the statistical model used to make predictions. Space-filling designs, in which the inputs are "spread out" across the entire design space, are a popular choice for initial designs (Kleijnen et al. 2005) because they maximize the possibility of identifying potential regions that contain the optimum when we have no prior information about the function. In this article, we use maximin LHS designs (Joseph and Hung 2008) because their projection properties provide useful information for performing variable selection.

Evaluating f at each row of $\mathbf{X}_0$ produces a response vector, $\mathbf{y}$, from the model $Y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$ where $\epsilon \sim N(0, \tau^2)$. A surrogate model, $\hat{f}$, is constructed from $\mathbf{y}$ and is used to make predictions for an arbitrary input $\mathbf{x}$. The surrogate model considered in this article assumes that f is a realization of a GP with mean function $E[f(\mathbf{x})] = \mu(\mathbf{x})$ and covariance function $\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = \sigma^2 K(\mathbf{x}, \mathbf{x}')$ for any two inputs $\mathbf{x}$ and $\mathbf{x}'$. Following Welch et al. (1992), we set $\mu(\mathbf{x}) \equiv \mu$ for all $\mathbf{x}$. There are numerous choices for correlation functions, including Matérn, nonstationary correlation functions, and BSS-ANOVA (Reich, Storlie, and Bondell 2009). Although a nonstationary correlation function could be more appropriate, they can require a large number of design points for proper estimation, which we cannot afford for our problem of interest. Instead, we choose the squared exponential correlation function (Sacks, Schiller, and Welch 1989)

$$K(\mathbf{x}, \mathbf{x}') = \exp\left\{-\sum_{k=1}^{p} \gamma_k \left(x_k - x_k'\right)^2\right\}, \tag{1}$$

where $\gamma_1, \ldots, \gamma_p \geq 0$ are the correlation range parameters. If $\gamma_k = 0$, then varying x_k across $[0, 1]$ has no effect on the response.

The covariance function for f induces a covariance function for $Y(\mathbf{x})$, which includes a nugget term τ^2 to account for random variation. Even for deterministic functions where $Y(\mathbf{x}) = f(\mathbf{x})$, including a nugget effect can protect against violations of model assumptions (Gramacy and Lee 2012). Letting $Y_i(\mathbf{x})$ denote the ith observation at $\mathbf{x}$, we then have

$$\text{cov}[Y_i(\mathbf{x}), Y_j(\mathbf{x}')] = \begin{cases} \sigma^2 + \tau^2 & \text{if } \mathbf{x} = \mathbf{x}' \text{ and } i = j \\ \sigma^2 K(\mathbf{x}, \mathbf{x}') & \text{otherwise.} \end{cases} \tag{2}$$

Let $\mathbf{V}_\mathbf{X}$ be the $n \times n$ covariance matrix of $\mathbf{y}$ from design matrix $\mathbf{X}$ and let $\mathbf{v}(\mathbf{x})$ be the $n \times 1$ vector of covariances between $\mathbf{y}$ and new observation $Y(\mathbf{x})$. The prediction for $f(\mathbf{x})$, conditional on $\mathbf{y}$, is a Gaussian random variable with mean and variance

$$\begin{aligned} \hat{f}(\mathbf{x} \mid \boldsymbol{\Omega}) &= \mu + \mathbf{v}(\mathbf{x})^T \mathbf{V}_\mathbf{X}{}^{-1} (\mathbf{y} - \mu \mathbf{1}_n), \\ s^2(\mathbf{x} \mid \boldsymbol{\Omega}) &= \sigma^2 - \mathbf{v}(\mathbf{x})^T \mathbf{V}_\mathbf{X}{}^{-1} \mathbf{v}(\mathbf{x}), \end{aligned} \tag{3}$$

where $\boldsymbol{\Omega}$ denotes the vector of GP parameters (Gelman et al. 2004). Of course, $\boldsymbol{\Omega}$ needs to be estimated from the available data, which we do following Linkletter et al. (2006) which incorporates global variable selection, described next.

2.2. Bayesian Estimation and Global Variable Selection

Each input variable x_k influences f through its corresponding range parameter γ_k in $K(\mathbf{x}, \mathbf{x}')$, where $\gamma_k = 0$ implies that the

input variable is globally inactive. Linkletter et al. (2006) placed positive mass on $\gamma_k = 0$ through a mixture prior such that

$$\gamma_k = u_k b_k, \qquad u_k \sim \text{Gamma}(a_u, b_u), \qquad b_k \sim \text{Bernoulli}(\theta), \tag{4}$$

where u_k is independent of b_k, and $\theta \sim \text{Beta}(a_\theta, b_\theta)$ is the probability of each variable being globally active. More details on parameter priors are available in Appendix A.1.

The decision to declare an input variable globally active is based on the posterior probability $\Pr(b_k = 1 \mid \mathbf{y}) = \Pr(\gamma_k > 0 \mid \mathbf{y}) \equiv \hat{b}_k$. Variable k is declared globally inactive if $\hat{b}_k < g$ where $g \in (0, 1)$ is some threshold. Following Linkletter et al. (2006), a data-driven estimate of g may be found by augmenting the design with one or more random inputs and setting g to be the estimated probability of those variables being active. In this article, once variable k is deemed globally inactive, the kth column of $\mathbf{X}$ is permanently removed from future consideration and the remaining GP parameters are re-estimated. Henceforth, p will always reference the current number of variables that are deemed globally active at the current sequential step.

Each posterior draw $\boldsymbol{\Omega}_t$, $t = 1, \ldots, M$, results in a new prediction surface $\hat{f}_t = \hat{f}(\cdot|\boldsymbol{\Omega}_t)$ and $\hat{\boldsymbol{\chi}}_t$, that is estimated from $\hat{f}_t$. We will also make use of the marginal prediction surface $\hat{f} = M^{-1} \sum_t \hat{f}_t$ and define the estimated global maximizer to be

$$\hat{\boldsymbol{\chi}} = \arg\max_{\mathbf{x}} \hat{f}\,. \tag{5}$$

Note this estimator may differ from the alternative estimator $M^{-1} \sum_t \hat{\boldsymbol{\chi}}_t$, the average of the maximizer posterior draws.

2.3. Specifying the Acquisition Function

To determine a new design point to help identify $\boldsymbol{\chi}$, Jones, Schonlau, and Welch (1998) introduced the efficient global optimization (EGO) algorithm, which balances exploring the design space and honing in on areas likely containing $\boldsymbol{\chi}$. As introduced by Močkus (1975), the improvement at any $\mathbf{x}$ is $I(\mathbf{x}) = \max\{f(\mathbf{x}) - y(\mathbf{x}_{\text{opt}}), 0\}$, where $\mathbf{x}_{\text{opt}}$ is the row of $\mathbf{X}$ where $y(\mathbf{x}_{\text{opt}})$ is the largest observed response in $\mathbf{y}$. Since f is unknown, the EGO algorithm instead uses $\hat{f}$ to compute the expected improvement, $\text{EI}(\mathbf{x}) = E[I(\mathbf{x})]$. Jones, Schonlau, and Welch (1998) showed that EI can be written as

$$\text{EI}(\mathbf{x}) = s(\mathbf{x}) \left\{ Z(\mathbf{x}) \Phi\left[Z(\mathbf{x})\right] + \phi\left[Z(\mathbf{x})\right] \right\}, \tag{6}$$

where $s(\mathbf{x}) = \sqrt{s^2(\mathbf{x})}$, $Z(\mathbf{x}) = [\hat{f}(\mathbf{x}) - y(\mathbf{x}_{\text{opt}})]/s(\mathbf{x})$, and $\Phi(\cdot)$ and $\phi(\cdot)$ are the CDF and PDF of a standard normal distribution, respectively. The next input is $\mathbf{x}^* \equiv \arg\max_{\mathbf{x}} EI(\mathbf{x})$.

The EGO algorithm was built for deterministic computer simulations, where $\tau^2 = 0$. The augmented EI criterion (Huang et al. 2006), or AEI, is more appropriate for nondeterministic functions

$$\text{AEI}(\mathbf{x}) \equiv E\left[\max\{\hat{f}(\mathbf{x}) - \hat{f}(\mathbf{x}_{\text{opt}}), 0\}\right]\left(1 - \frac{\tau}{\sqrt{s^2(\mathbf{x}) + \tau^2}}\right), \tag{7}$$

where $\mathbf{x}_{\text{opt}} \equiv \arg\max_{\mathbf{x}_i \in \mathbf{X}}\{\hat{f}(\mathbf{x}_i) - \nu s(\mathbf{x}_i)\}$ for a given $\nu \geq 0$. Huang et al. (2006) stated that the $\mathbf{x}_{\text{opt}}$ design point is chosen to reflect the user's degree of risk aversion, where $\nu = 1$ represents a "willingness to trade 1 unit of predicted objective value for 1 unit of the standard deviation of prediction uncertainty." See Brochu et al. (2010) for a discussion of other acquisition functions for identifying $\boldsymbol{\chi}$.

There are numerous algorithms to optimize AEI. Picheny and Ginsbourger (2014) optimized AEI through genetic optimization with derivatives, developed by Mebane and Sekhon (2011). Kleijnen (2015) constructed a space-filling design of candidate points $\mathbf{C} \subset [0, 1]^p$ and sets the next design point to be $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathbf{C}} \text{AEI}(\mathbf{x})$. These approaches are not immediately appealing for the problem at hand because (1) they may fail to find the true maximizer of AEI due to its multimodal nature in moderate to high dimensions and (2) they may encourage initial exploration of the design space, leading to poor initial improvement over the current $\hat{\boldsymbol{\chi}}$. Sections 3 and 4 describe how we address these issues using a local variable selection algorithm and adaptive candidate sets to improve optimization of AEI and f.

3. Bayesian Local Variable Selection

Optimizing AEI around the current $\hat{\boldsymbol{\chi}}$ is appealing for multiple reasons. For one, it limits the possibility of the next design point to explore unobserved regions of the design space having high uncertainty under the surrogate model and instead encourages identification of a local optimum in an area that has been estimated to contain $\hat{\boldsymbol{\chi}}$. Hence, it is more likely to lead to an updated $\hat{\boldsymbol{\chi}}$ with a larger $f(\hat{\boldsymbol{\chi}})$ than if we chose a design point by globally optimizing AEI. Another reason is that, even when a variable is determined to be globally active, and hence has $\hat{\gamma}_k > 0$, it may be that the variable is not important in a localized region of interest. Employing local variable selection can help to optimize AEI and update $\hat{\boldsymbol{\chi}}$ by significantly reducing the dimensionality of the optimization problem. This localized strategy would be especially beneficial for expensive functions with a potentially limited number of additional evaluations.

We define here a new measure of local importance defined on some region of the input space and, in the next section, develop a flexible algorithm that uses local variable selection to identify the maximizer for AEI. An appealing aspect of the proposed measure is its avoidance of expensive gradient evaluations. To motivate the measure, consider the two-dimensional toy example in Figure 1.

Both x_1 and x_2 are needed to describe the function globally, but there are areas that would require only one of the variables for optimization. Focusing on the localized, rectangular region labeled as (b), we make a baseline predicted surface using our current posterior estimates of γ_1 and γ_2.

The global parameter γ_2 is likely greater than 0, but x_2 clearly does not substantially affect f in this region. Consider the alternative predicted surface restricted to this region, where γ_2 is temporarily set to 0 and γ_1 is the same as in the baseline surface. If this alternative predicted surface is similar to the baseline predicted surface, we would conclude that x_2 is locally inactive. Figure 2 shows the baseline and alternative predicted surfaces for this rectangular region and demonstrates how one would reach the conclusion that x_2 is locally inactive.

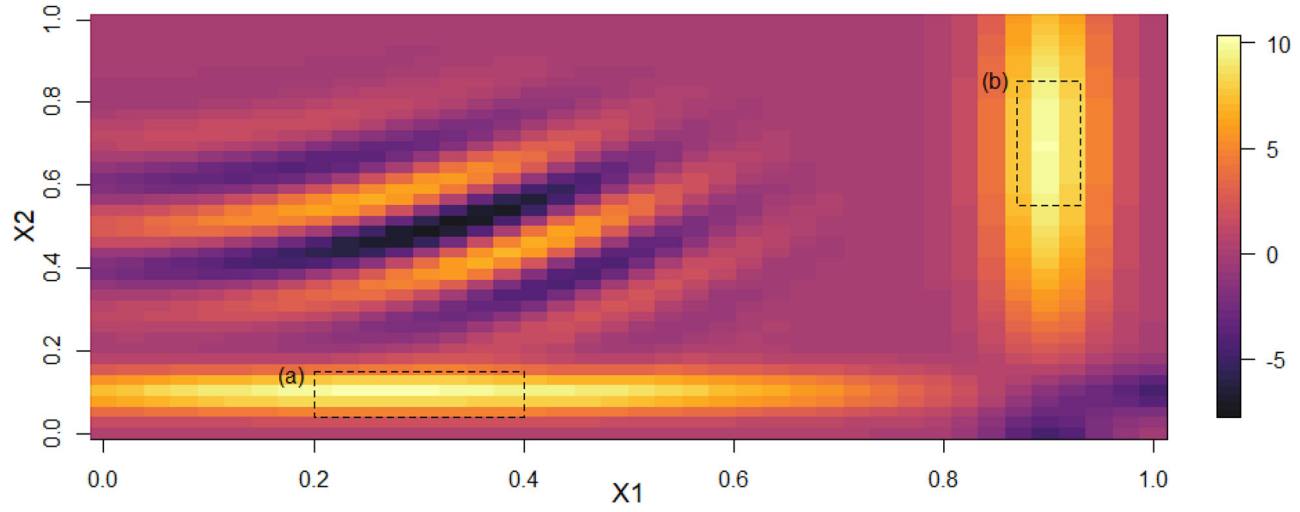

Figure 1. Toy example function $f(x_1, x_2)$ in $[0, 1]^2$ having two local regions in which f attains its maximum. In regions (a) and (b), optimization only needs to be done with respect to x_2 and x_1, respectively.

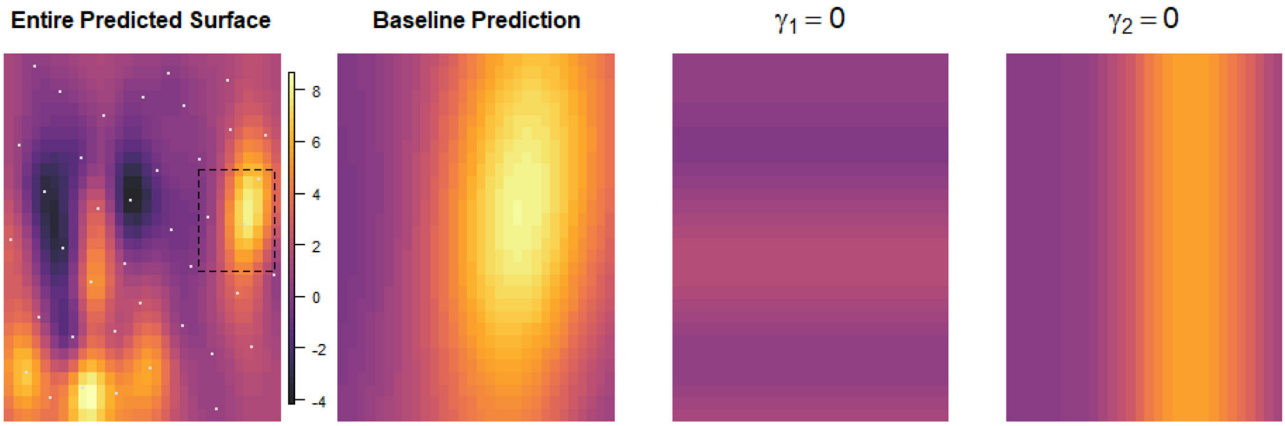

Figure 2. The far left image is $\hat{f}$ for the Figure 1 function with a highlighted region of interest, shown in greater detail in the second plot. This represents the "baseline" predicted surface for determining local activity. The third and fourth plots show the alterative predicted surfaces: $\hat{f}^1$ with $\gamma_1 = 0$ and $\hat{f}^2$ with $\gamma_2 = 0$. The similarity between the baseline surface and $\hat{f}^2$ indicates that x_2 is locally inactive in that region.

Our approach assesses local variable importance within a neighborhood of each maximizer posterior draw $\boldsymbol{\chi}_t$ by comparing the baseline predicted surface, $\hat{f}_t$ to each of the p alternative predicted surfaces, denoted by $\hat{f}_t^k$, generated by temporarily fixing $\gamma_k = 0$. To this end, we first generate q prediction points $\mathbf{Q}_t$ from a truncated multivariate normal distribution

$$\mathbf{Q}_t \sim TN_{[0,1]^p}(\hat{\boldsymbol{\chi}}_t, \delta \mathbf{I}), \tag{8}$$

where δ controls how far the prediction points are spread from $\hat{\boldsymbol{\chi}}_t$, and the truncation keeps $\mathbf{Q}_t$ within the $[0, 1]^p$ design space. We calculate the baseline and alternative predictions at the points in $\mathbf{Q}_t$, denoted $\hat{f}_t(\mathbf{Q}_t)$ and $\hat{f}_t^k(\mathbf{Q}_t)$, respectively. We compare the baseline and alternative predictions using the squared correlation

$$R_{kt}^2 = \text{corr}\left(\hat{f}_t(\mathbf{Q}_t), \hat{f}_t^k(\mathbf{Q}_t)\right)^2. \tag{9}$$

If R_{kt}^2 is close to one, then setting $\gamma_k = 0$ did not greatly affect the predictions, offering evidence that x_k is locally inactive.

It is possible that the $\hat{\boldsymbol{\chi}}_t$'s will be dispersed across the input space and different variables are likely to be locally active with respect to different $\hat{\boldsymbol{\chi}}_t$ (Bai et al. 2014). Anticipating this possibility, we average over the R_{kt}^2 values and define the local importance L_k of input k as

$$L_k \equiv 1 - \text{mean}(R_{k1}^2, \ldots, R_{kM}^2). \tag{10}$$

Then L_k is an averaged measure of local importance across the posterior distribution of $\boldsymbol{\chi}$. We declare a variable to be locally active if $L_k \geq \rho$ for $0 < \rho < 1$ and let $\mathbf{A}$ denote the set of locally active variables. Algorithm 1 summarizes this procedure, including an additional step to perform the above calculations on only $m < M$ of the posterior draws for computational reasons. The choice of the m draws should be done carefully to be representative of the entire posterior distribution.

Declaring a variable to be locally active does not necessarily mean the function exhibits nonstationary behavior. For a function generated from a stationary GP, the parameters γ_k describe the correlation with respect to the entire input space. As we focus our attention to a smaller region of interest, variables having $\gamma_k > 0$ will start to appear unimportant. The larger γ_k is, the smaller the region needs to be for this to happen. We allow the uncertainty of $\boldsymbol{\chi}$ to dictate the size of the region. Even if f is generated from a nonstationary GP, our use of a surrogate

Algorithm 1 Identifying locally active variables

Initialize ρ and δ; randomly sample $m \leq M$ posterior draws
for $t \in \{1, \ldots, m\}$ **do**
  Estimate $\hat{\boldsymbol{\chi}}_t$ using $\boldsymbol{\Omega}_t$ and all globally active variables
  Construct q prediction points $\mathbf{Q}_t$ centered around $\boldsymbol{\chi}_t$
  Determine baseline predictions, $\hat{f}$ at $\mathbf{Q}_t$ using $\boldsymbol{\Omega}_t$
  for variable $k \in \{1, \ldots, p\}$ **do**
    Make alternative predictions, $\hat{f}_t^k$ at $\mathbf{Q}_t$ and calculate R^2_{kt}
Calculate $L_k = 1 - \text{mean}(R^2_{k1}, \ldots, R^2_{km})$.
return $\mathbf{A} = \{k : L_k \geq \rho \mid \hat{\boldsymbol{\chi}}\}$ the set of locally active variables

function assuming a stationary GP is necessary given our cost assumptions of evaluating f. To further support this, our toy example and numerical studies involve functions that exhibit nonstationary behavior.

4. Sequential Optimization Using SOLID

In practice, finding the optimal AEI often involves reducing the optimization space, such as with trust regions. If $\hat{\boldsymbol{\chi}}$ is near $\boldsymbol{\chi}$, further exploration would be unnecessary, and restricting the search for the AEI maximizer to a small neighborhood of $\hat{\boldsymbol{\chi}}$ would be advantageous. Local variable selection can further reduce the dimension of this localized search space. We detail here the SOLID algorithm that uses global and local variable importance measures to improve estimation of $\boldsymbol{\chi}$.

In SOLID, rather than restrict the AEI search space for the kth locally active variable to be within some neighborhood centered at $\hat{\boldsymbol{\chi}}$, we restrict the space to be

$$\mathcal{R}_k^\delta := \left[\min\left(\hat{\chi}_{k,1}, \ldots, \hat{\chi}_{k,m}\right) - \delta \,, \max\left(\hat{\chi}_{k,1}, \ldots, \hat{\chi}_{k,m}\right) + \delta \right] \cap [0, 1] \,, \tag{11}$$

using the kth coordinate of the m $\hat{\boldsymbol{\chi}}_t$'s and δ implemented in Algorithm 1. For the jth locally inactive variable, we set $\mathcal{R}_j^\delta := \hat{\chi}_j$, the jth coordinate of $\hat{\boldsymbol{\chi}}$ which is calculated from $\hat{f}$. Let $\mathcal{R}^\delta$ denote the corresponding restricted search space.

Unlike the trust region in Regis (2016), the range of the $\mathcal{R}^\delta$ search is guided directly by the estimates of $\hat{\chi}_{k,t}$ and explores only the locally active variables. Moreover, (11) incorporates the uncertainty of $\boldsymbol{\chi}$ into the search space $\mathcal{R}^\delta$. The inclusion of the δ parameter allows us to further expand the region if the distribution of the $\hat{\chi}_{k,1}, \ldots, \hat{\chi}_{k,m}$ may be too narrow (perhaps due to selecting a smaller m for better computational performance). One could also use a different parameter than the δ used in Algorithm 1.

To search for the AEI optimum in $\mathcal{R}^\delta$, we construct an $|\mathbf{A}|$-dimensional maximin LHS design $\mathbf{L}_\delta \subset \mathcal{R}^\delta$ with c settings to evaluate AEI. For example, if only the first $a < p$ variables are locally active, then the set of restricted candidate points $\mathbf{C}_\delta \subseteq \mathcal{R}^\delta$ would be

$$\mathbf{C}_\delta = \begin{bmatrix} \mathbf{L}_\delta & \hat{\chi}_{a+1}\mathbf{1}_c & \ldots & \hat{\chi}_{p-1}\mathbf{1}_c & \hat{\chi}_p\mathbf{1}_c \end{bmatrix}. \tag{12}$$

It is possible that $\mathcal{R}^\delta$ is still too restrictive so we also consider a slightly larger space $\mathcal{R}^{\mathbf{A}}$ with $\mathcal{R}_k^{\mathbf{A}} := [0, 1]$ for each locally active variable k, and $\mathcal{R}_j^{\mathbf{A}} := \hat{\boldsymbol{\chi}}_j$ for each locally inactive variable j. This allows us to consider exploration of unobserved locations, but only within the locally active dimension. We again use a maximin LHS design $\mathbf{L}$ of dimension $|\mathbf{A}|$, within $\mathcal{R}^{\mathbf{A}}$ for the c candidate points. These unrestricted candidate points $\mathbf{C}_{\mathbf{A}}$ are constructed in the same manner as in (12), where the column of each locally inactive variable $j \in \mathbf{A}^c$ is $\hat{\chi}_j\mathbf{1}_c$. Note that $\mathcal{R}^\delta \subseteq \mathcal{R}^{\mathbf{A}}$ but that $\mathcal{R}^\delta$ is more densely concentrated around the $\hat{\boldsymbol{\chi}}_t$'s.

While it is possible to combine both $\mathbf{C}_\delta$ and $\mathbf{C}_{\mathbf{A}}$ into one large candidate set, we have found that the candidate points with the greatest AEI often all reside in one of the two sets. Whichever set has the largest AEI becomes the final set of candidate points $\mathbf{C}$. Conceptually, this helps us see if SOLID is honing-in on a restricted space $\mathcal{R}^\delta$ or exploring the larger space $\mathcal{R}^{\mathbf{A}}$. Using the $|\mathbf{A}|$-dimensional gradient of AEI (see Picheny and Ginsbourger 2014), we conduct line searches from the five most promising candidates in $\mathbf{C}$, restricting the search to lie within a ball of radius δ (as specified in (8)). After the line searches are complete, the one with the largest AEI is chosen as the next design point.

Thus far, we have described using the local variable selection results only for optimizing AEI, but they could also apply for the estimate of $\hat{\boldsymbol{\chi}}$. One may be skeptical of doing this since the proposed local variable selection procedure uses the posterior draws $\hat{\boldsymbol{\chi}}_t$ which are calculated without local variable selection. By estimating $\boldsymbol{\chi}$ across all globally active variables, we are likely to observe larger variation of $\boldsymbol{\chi}_t$ because we are optimizing in higher dimensions. Modifying the $\hat{\boldsymbol{\chi}}$ optimization space to $\mathcal{R}^{\mathbf{A}}$ should provide a more stable estimator.

We have described, up to this point, a single iteration of a sequential optimization algorithm detailing the global and local variable selection procedure, the localized AEI optimization, and the localized estimation of $\boldsymbol{\chi}$. The next iteration starts with evaluation of the recommended design point from the localized AEI optimization. We then re-estimate the GP parameters using MCMC with priors from the initial iteration. Recall, if at least one variable is deemed globally inactive at this next step then we would again re-estimate the GP parameters and perform global variable selection using only the globally active variables. Next we perform our Bayesian local variable selection across all globally active variables and maximize AEI in the new localized region. Note that previous results of the local variable selection algorithm are ignored. This way, any misclassification of locally active/inactive variables will not have long-lasting consequences. This flexibility also allows assessment of local importance to recalibrate when $\hat{\boldsymbol{\chi}}$ changes. The SOLID procedure is summarized in Algorithm 2.

We demonstrate SOLID in Figure 3 on the toy function (Figure 1) that includes a third, unimportant variable x_3. We set the global and local thresholds to be $g = 0.50$ and $\rho = 0.30$, and set $\delta = 0.15$. We considered $c = 300$ candidate points when optimizing AEI. Observations were generated with noise, $\tau^2 = 0.08$. For simplicity, the marginal surfaces were built using $m = 25$ random draws using MCMC chains of length $M = 500$. We start with an initial maximin LHS design with $n_0 = 10, p = 3$, shown as ▲ in Figure 3 in the upper left panel.

In the first iteration, $\mathbf{y}(\mathbf{x}_{\text{opt}}) = 7.71$ and all three variables were deemed globally active. Local importance was assessed

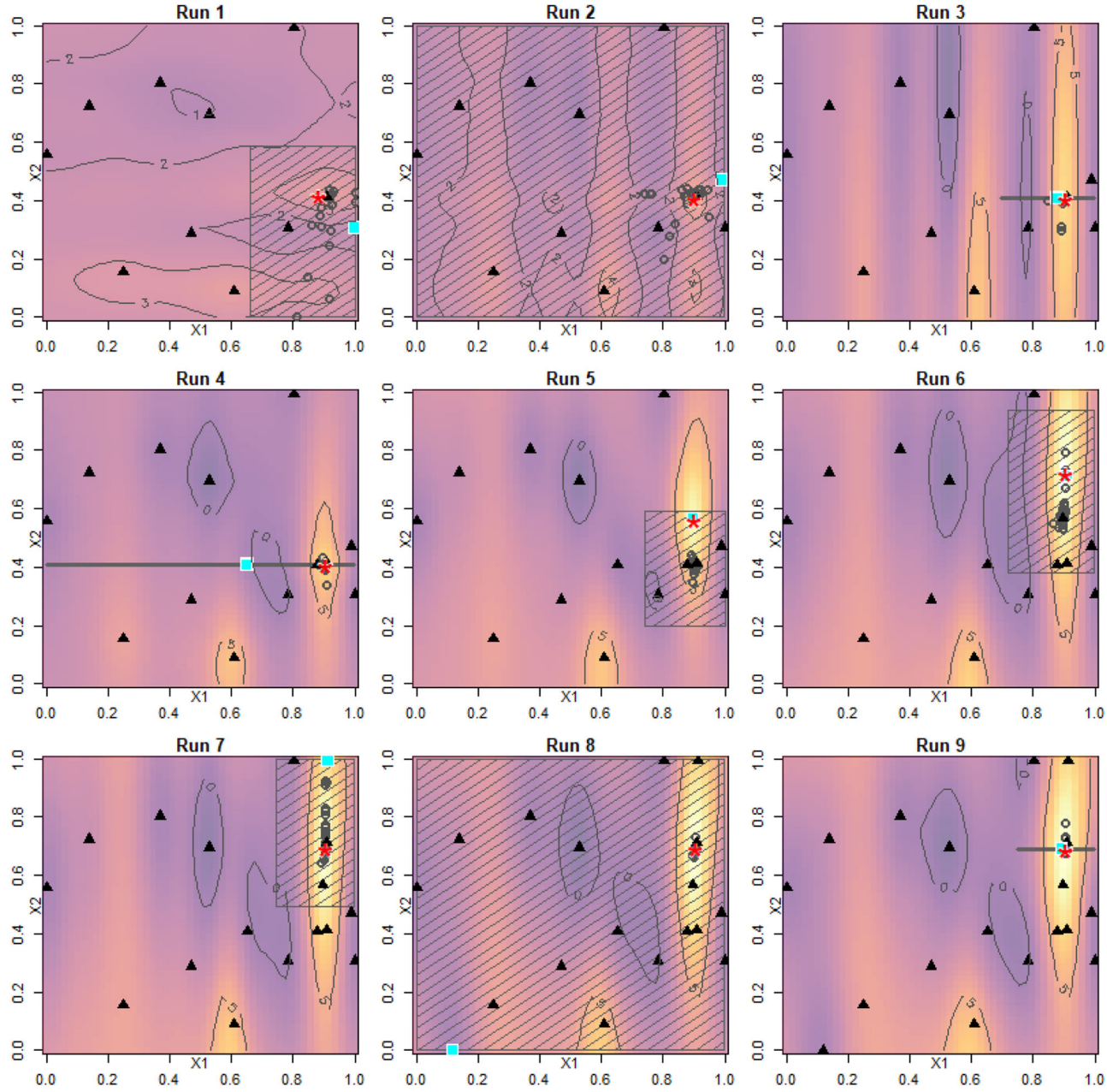

Figure 3. The predicted surface $\hat{f}$ of the Figure 1 function across nine runs of SOLID. The design points (▲) and $\hat{\boldsymbol{\chi}}$ (∗) are shown. Local importance is assessed at each of the 25 posterior draws of $\hat{\boldsymbol{\chi}}_t$ (◎). The shaded rectangles represents either $\mathcal{R}^{\mathbf{A}}$ or $\mathcal{R}^{\delta}$, depending on which has the AEI maximizer (■). In runs 3, 4, and 9, x_2 was found to be locally inactive, so the candidate points explore only the x_1 dimension (horizontal line).

around the $m = 25$ posterior draws of $\hat{\boldsymbol{\chi}}_t$, shown as open circles in Figure 3. All three variables were also deemed locally active, with $L_1 = 0.52, L_2 = 0.81$, and $L_3 = 0.76$. It follows then that $\mathcal{R}^{\mathbf{A}} = [0,1]^3$, the entire input space, and $\mathcal{R}^{\delta}$ (visualized in just the important dimensions as the shaded rectangle in the upper left panel of Figure 3) was bounded by $0.66 \leq x_1 \leq 1.00, 0.00 \leq x_2 \leq 0.59$ and $0.44 \leq x_3 \leq 0.94$. The localized optimum estimation step was not technically localized since $\mathcal{R}^{\mathbf{A}} = [0,1]^3$ and we determined $\hat{\boldsymbol{\chi}}^0 = (0.88, 0.42, 0.57)$.

Next we optimized AEI to determine the next design point. The $c = 300$ candidate points were generated in $\mathcal{R}^{\mathbf{A}}$ and $\mathcal{R}^{\delta}$, producing $\mathbf{C}^{\mathbf{A}}$ and $\mathbf{C}^{\delta}$, respectively. The set $\mathbf{C}_{\delta}$ contained the point with the largest AEI and a line search optimization algorithm contained in this restricted space determined the next design point to be $\mathbf{x}^* = (1.00, 0.31, 0.67)$. This point was added to $\mathbf{X}$ and we then evaluated $Y(\mathbf{x}^*)$.

In the second run, all variables were again found to be globally and locally active. The updated optimum was $\hat{\boldsymbol{\chi}}^1 = (0.90, 0.40, 0.58)$. Here the unrestricted candidate set $\mathbf{C}_{\mathbf{A}} = [0,1]^3$ was preferred for AEI optimization and the next selected point was $\mathbf{x}^* = (0.99, 0.47, 0.55)$, close to $\hat{\boldsymbol{\chi}}^1$.

In the third run, we found $\hat{b}_3 = 0.49 < g = 0.5$ so variable x_3 was permanently removed and the remaining GP parameters were re-estimated. Both x_1 and x_2 were still deemed globally active, as they should be. Their respective local importance measures were $L_1 = 0.85$ and $L_2 = 0.29$. With $L_2 < \rho = 0.3$, x_2 was declared locally inactive, and so $\mathcal{R}^{\mathbf{A}}$ and $\mathcal{R}^{\delta}$ were entirely contained within the x_1 dimension, both fixing x_2 at

Algorithm 2 Summary of SOLID

1: Set n_0, N (maximum number of evaluations), g, δ, ρ, M, m, c
2: Create an initial maximin LHS(n_0, p) design, $\mathbf{X}$
3: Generate $\mathbf{y}$ from $Y(\mathbf{X})$
4: **for** step $i \in \{0, \ldots, N\}$ **do**
5: Obtain M posterior draws of $\boldsymbol{\Omega}_t$ and $\boldsymbol{\chi}_t$ (Section 2.2); calculate $\hat{f}$ and $\hat{\boldsymbol{\chi}}$.
6: **Global variable selection**: Remove variables with $\hat{b}_k < g$ from $\mathbf{X}$; if variables removed, repeat step (5) with new $\mathbf{X}$
7: **Local variable selection**: Implement Algorithm 1 with δ, ρ, and $m < M$ $\boldsymbol{\chi}_t$'s; store $\mathbf{A}$
8: Define restricted $\mathcal{R}^{\delta}$ and unrestricted $\mathcal{R}^{\mathbf{A}}$ search spaces
9: **Localized optimum estimation**: Update estimate $\hat{\boldsymbol{\chi}}$ in $\mathcal{R}^{\mathbf{A}}$ using $\hat{f}$; store as $\hat{\boldsymbol{\chi}}^i$.
10: Create maximin LHS designs $\mathbf{C}_{\delta} \subseteq \mathcal{R}^{\delta}$ and $\mathbf{C}_{\mathbf{A}} \subseteq \mathcal{R}^{\mathbf{A}}$
11: Evaluate AEI in $\mathbf{C}_{\delta}$ and $\mathbf{C}_{\mathbf{A}}$; define the set with the largest AEI as $\mathbf{C}$
12: **Localized AEI estimation**: Perform line search optimization to identify $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathbf{C}} AEI(\mathbf{x})$
13: Augment $\mathbf{x}^*$ to $\mathbf{X}$; generate $Y(\mathbf{x}^*)$ and add to $\mathbf{y}$
14: **return** $\{\hat{\boldsymbol{\chi}}^0, \ldots, \hat{\boldsymbol{\chi}}^N\}$

0.41. Maximizing AEI, the next design point $\mathbf{x}^* = (0.88, 0.41)$ was added to $\mathbf{X}$. Note that the setting for the third input variable was no longer considered. If a value for that variable were required for the function to be evaluated, one could choose the corresponding coordinate from the previous optimum estimate, which in this case was 0.58 for x_3.

For the remaining runs, x_1 and x_2 were always found to be globally active but x_2 was found locally inactive in runs 4 and 9. As design points were added and parameters were updated, $f(\hat{\boldsymbol{\chi}})$ rose steadily: 7.42, 9.41, 9.89, 9.92, 9.93, and 10.00, with 10.00 being the largest possible value.

5. Simulation Study

We conducted a simulation study to evaluate the effects of global and local variable selection on sequential optimization. We compared four approaches: (1) GVS conducts global variable selection only; (2) SOLID, as described in Section 4; (3) Oracle uses only the known globally active variables (without performing any variable selection); and (4) None uses all variables. Within each simulation run, all four approaches used the same initial design, a maximin Latin hypercube design, and the same vector of initial responses. The responses were measured with error, $\tau^2 = 0.05$.

We compared results for three different test functions in $p = 15$ dimensions, named Beach, Drum, and Simba. Although these are not conventionally high-dimensional functions, they are still sufficiently large enough to be important for practical concerns. All three functions have 6 truly globally active variables. The names for each function come from their visualization in the x_1 and x_2 subspace with all other variables set to χ_j, their values in the global maximizer, visualized in the top row of Figure 4. The Beach function resembles a sandy beach along a pink sea; Drum resembles an oval shaped drum; and Simba is reminiscent of the scene from Disney's "The Lion King" (Hahn, Allers, and Minkoff 1994), where Rafiki holds Simba high up on Pride Rock, against the rolling hills and surrounding plains. The Beach function is constructed to have a local mode in a region far away from $\boldsymbol{\chi}$. Four variables are locally active around this local mode, but only x_1, x_2 and x_3 are locally active around $\boldsymbol{\chi}$. The Drum function is primarily influenced by x_3, but around $\boldsymbol{\chi}$, x_1 through x_5 are all locally active. The Simba function is especially challenging to optimize, since it has a large number of local modes involving all six globally active variables. Around $\boldsymbol{\chi}$, however, only x_1, x_2 and x_3 are locally active. Results for three additional test functions commonly found in the optimization literature are shown in Section 3 of the supplementary materials.

It was important to choose n_0 to be large enough to construct a reasonable $\hat{f}$ without being so large that $\boldsymbol{\chi}$ would easily be known. To that end, the initial designs for Beach and Drum had $n_0 = 70$ observations, and Simba had $n_0 = 80$. For all methods we used noninformative priors $\sigma_{\mu} = 100$, $a_{\eta} = b_{\eta} = 0.1$, $a_{\theta} = b_{\theta} = 1$. We set $a_u = 1$ and $b_u = 10$ so that $E(u_k) = 10$ and $\text{var}(u_k) = 100$ (see (4)). We ran MCMC chains of length $M = 1000$, of which $m = 100$ posterior draws were used for the marginal surfaces. We set $\delta = 0.30$ and chose conservative global and local variable selection thresholds, $g = 0.05$ and $\rho = 0.02$. Of primary interest was determining the response of the true function f evaluated at $\hat{\boldsymbol{\chi}}$ across $N = 25$ additional runs. Because each initial designs gave a different $\hat{\boldsymbol{\chi}}^0$, our performance metric was relative improvement

$$f(\hat{\boldsymbol{\chi}}^i) - f(\hat{\boldsymbol{\chi}}^0). \tag{13}$$

To summarize across all $N = 25$ runs, we defined overall improvement $\frac{1}{25}\sum_{i=1}^{25} f(\hat{\boldsymbol{\chi}}^i) - f(\hat{\boldsymbol{\chi}}^0)$.

Averaging across 100 simulated initial designs, we present the mean relative improvement for each added sequential design point, for each approach and test function in Figure 5. As expected, Oracle performed the best on Beach and Drum, largely because it optimized across only the 6 globally active variables. Across all test functions, None performed the worst, since it always optimized over a 15-dimensional space. SOLID had higher mean relative improvement than GVS for each of the first 10 runs on the Drum, and each of the first 20 runs on the Beach. For the Simba function, SOLID had higher mean improvement than GVS and Oracle for each of the first 6 runs. In Table 1, SOLID had significantly higher overall improvement than GVS (p-value < 0.001) in terms on Beach and Simba, and even outperformed Oracle on Simba ($p = 0.003$).

SOLID was able to achieve its enhanced performance, not only by permanently removing variables through global variable selection, but by honing in on more promising lower-dimensional subspaces. Figure 6 shows that our proposed measure of local importance successfully captured the locally active variables. Table 2 shows that across all three test functions, SOLID was optimizing over fewer variables than GVS, as defined by the number of variables explored by the AEI function.

We also compared the methods in terms of computational costs. Oracle, which knows the set of globally active variables and does not perform any variable selection, took 1.8 hr to add 25 design points, averaged across all test functions. For every hour that Oracle took to obtain 25 new design points,

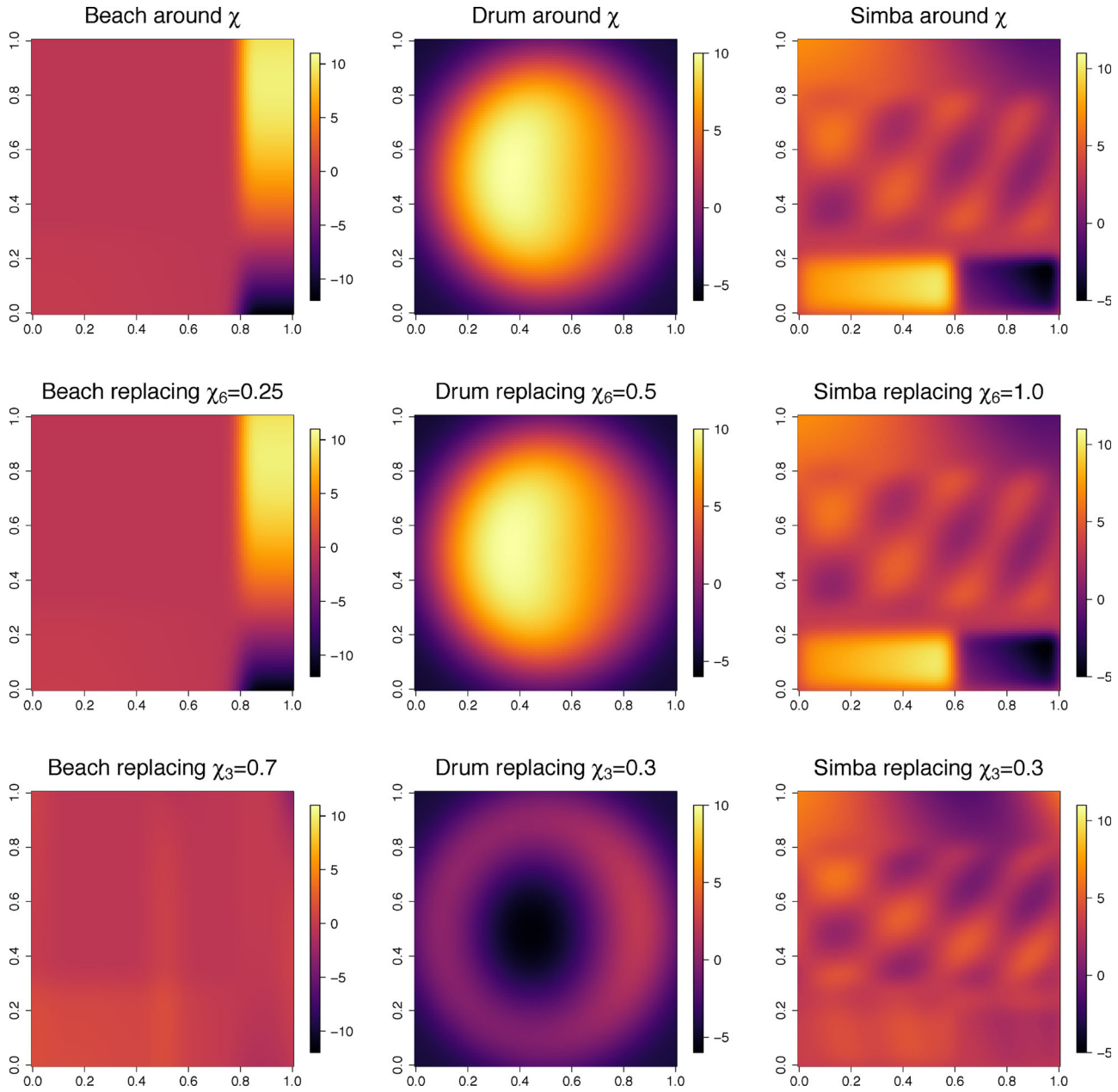

Figure 4. Each column shows a different six-dimensional test function in the x_1, x_2 dimension with the other coordinates fixed. The first row sets x_3 to x_6 to their optimal values in $\boldsymbol{\chi}$. Beach has $\boldsymbol{\chi} = 1, 0.85, 1, 0, 0, 0)$; Drum has $\boldsymbol{\chi} = (0.368, 0.533, 0, 1, 0.555, 1)$; and Simba has $\boldsymbol{\chi} = (0.523, 0.0999, 0, 0.298, 0.298, 0.245)$. Plots in the second row are nearly identical to those in the first, indicating x_6 is locally inactive. The third row shows that x_3 is locally active, as changing x_3 led to very different behavior of f.

Table 1. The mean improvement for all approaches and test functions averaging across 100 simulated initial designs and 25 runs.

	Oracle		SOLID		GVS		None		GVS ≠ SOLID
Test Function	Mean	SE	Mean	SE	Mean	SE	Mean	SE	*p*-value
Beach	1.49	0.026	1.30	0.023	1.15	0.024	1.00	0.021	<0.001
Drum	3.20	0.037	2.66	0.038	2.62	0.038	2.14	0.035	0.122
Simba	1.25	0.025	1.39	0.029	1.19	0.025	1.13	0.024	<0.001

NOTE: A Wilcoxon rank sums test shows that SOLID performs significantly better than GVS on the Beach and Simba functions.

None took 1.4 hr, GVS took 4.0 hr, and SOLID took 5.3 hr. Although computationally more expensive, if each evaluation of f is expensive, SOLID would still be preferable to GVS and None, since it required fewer evaluations of f to obtain equivalent or better estimates of $\boldsymbol{\chi}$. To see this, we compared the mean improvement value that Oracle achieves after 7 runs with the number of runs the other approaches needed to achieve at least that value (see Figure 5). On the Beach function, SOLID needed 10 runs, whereas GVS required 13, and none required 15. Similar patterns held for the Drum and Simba functions.

As evidenced in Figure 5, the mean improvement value for GVS eventually met (on the Beach and Simba functions) or

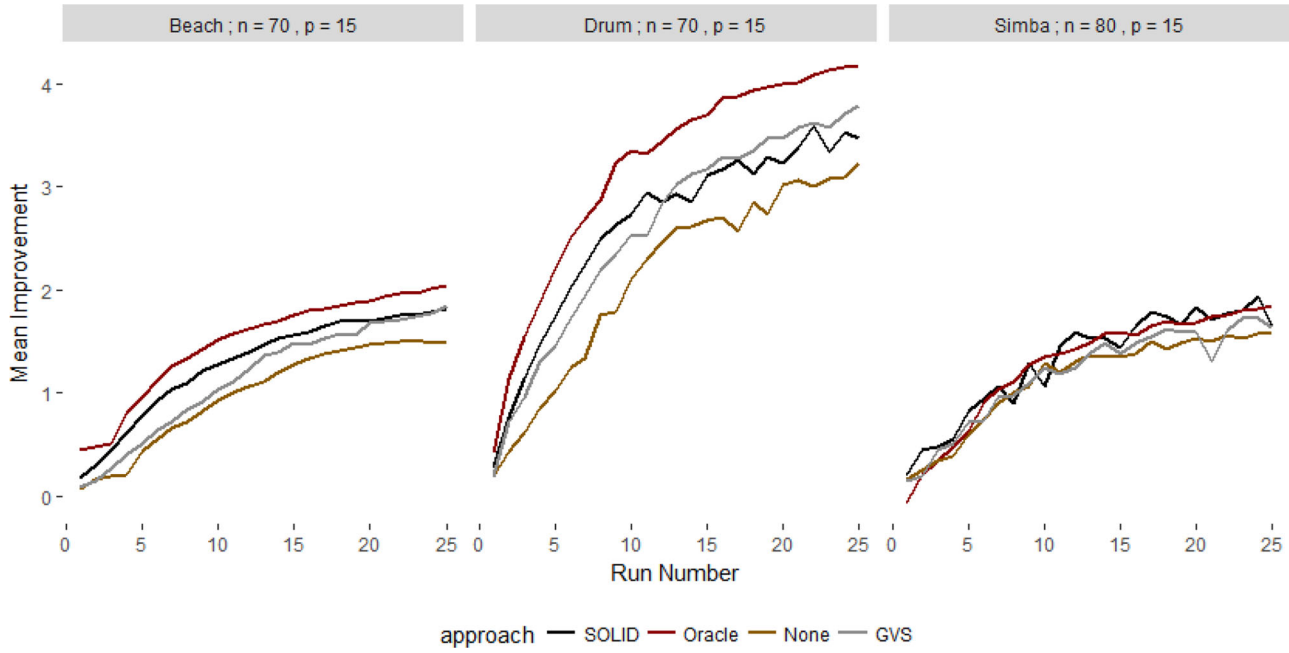

Figure 5. Mean relative improvement $(f(\hat{\boldsymbol{x}}^i) - f(\hat{\boldsymbol{x}}^0))$ across 100 simulations, using the three test functions for sequential runs $i \in \{1, \ldots, 25\}$. Improvement at run 0 is not shown as all approaches have zero relative improvement.

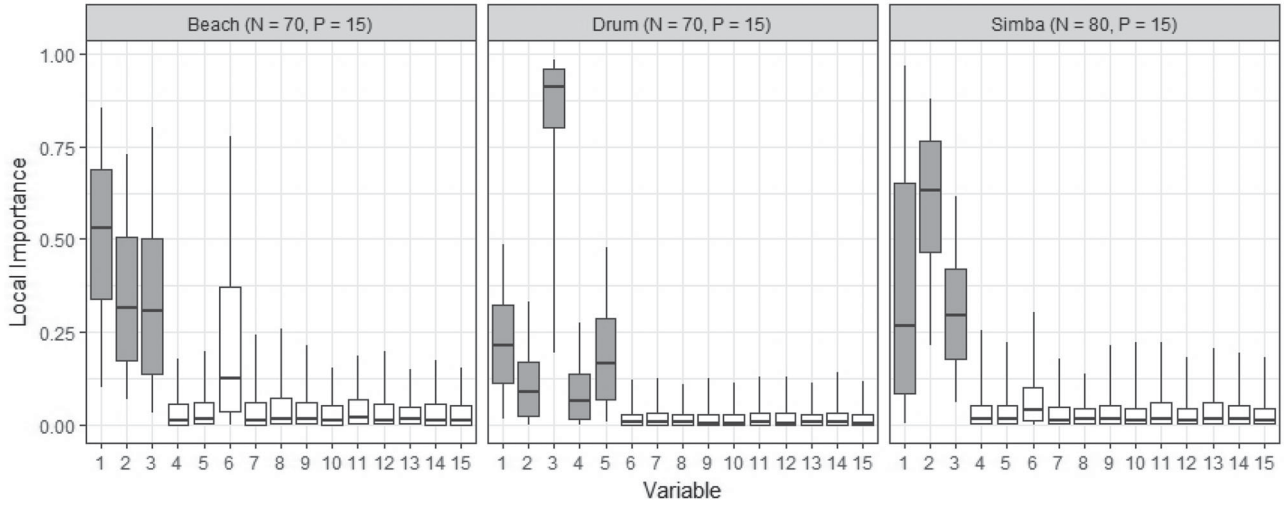

Figure 6. L_k boxplots over all 100 simulated datasets and 25 runs, where the "whiskers" are the 5th and 95th percentiles. Gray boxplots correspond to the truly locally active variables.

Table 2. The false positives are variables that are included in the design but are truly globally inactive.

	Mean number of variables at run 25 (across 100 simulations)					
Test function	Used for optimization			False positives		
approach	Beach	Drum	Simba	Beach	Drum	Simba
Oracle	6.00	6.00	6.00	0.00	0.00	0.00
SOLID	6.32	5.58	6.65	7.11	6.88	7.71
GVS	9.38	11.83	10.85	4.30	6.22	5.71
None	15.00	15.00	15.00	9.00	9.00	9.00

NOTE: There are 15 total variables; at most 9 are false positives.

exceeded (on the Drum function) the value obtained by SOLID. The difference in how these two methods selected inputs at which to evaluate f next could explain this result. By selecting inputs whose values vary in all p globally active dimensions, GVS may be better able to identify truly globally inactive variables and correctly remove them from the design matrix. By removing more globally inactive variables than SOLID, GVS could eventually experience comparable or better mean improvement values. In the first few sequential runs, however, the results favored SOLID.

6. Analysis of Sarcos Robot Data

The Sarcos robot dataset (Vijayakumar and Schaal 2000) consists of $n = 44{,}484$ observations and $p = 21$ input variables, available at *www.gaussianprocess.org/gpml/data*. The input variables are the positions, velocities, and accelerations of seven different points on a robot arm as it draws a figure eight (Vijayakumar, D'Souza, and Schaal 2005). We transform the inputs such that $\mathbf{x} \in [0,1]^{21}$. The response variable $Y(\mathbf{x})$ is the first of seven joint torque measurements (Parker 2015). Sequential optimization requires being able to evaluate the response surface at arbitrary input values, but this is not possible with the discrete Sarcos data. Therefore, for illustration purposes, we generated data assuming the true response surface is a kernel smoothed function. For any input $\mathbf{x}$, we have

$$f(\mathbf{x}) = \frac{\sum_{i\in\mathbb{S}} y(\mathbf{x}_i)K(\mathbf{x}_i,\mathbf{x})}{\sum_{i\in\mathbb{S}} K(\mathbf{x}_i,\mathbf{x})}, \tag{14}$$

where $\mathbb{S} = \{1,\ldots,n\}$, $\mathbf{x}_i$ are the observed inputs in the Sarcos dataset, and the kernel smoother is $K(\mathbf{x},\mathbf{x}') = \exp\{-\sum_{j=1}^{p} h^{-2}(x_j - x_j')^2\}$. Based on 5-fold cross-validation minimizing the out-of-sample prediction MSE, the best bandwidth was $h = 0.08272$.

With the `fields` package in R, we randomly selected inputs that led to space-filling designs. We included ten times as many initial design points as dimensions (Loeppky, Sacks, and Welch 2009). We considered only 15 sequential evaluations in this analysis due to the computational demands. Using only the GVS and SOLID approaches, we evaluated the improvement (13) at each run $i \in \{1,\ldots,15\}$. We set $g = 0.15$ and $\rho = 0.01$ to provide a moderate amount of variable selection, and we set $\delta = 0.20$ to emphasize local searches. $\tau^2 = 0.05$. We used the same priors and number of MCMC chains and posterior samples as in Section 5.

We present results for 100 simulated datasets in Figure 7. SOLID achieved greater improvement than GVS over 100 simulated datasets at nearly every run of the sequential design. Comparing overall improvement, SOLID was significantly better than GVS (p-value < 0.001). SOLID consistently used fewer variables for optimization than GVS. We found that neither method removed any variables based on global variable selection. However, by the final run, SOLID used 15.88 variables during its optimization and design selection, compared to 21 for GVS. Figure 7 shows the proportion of datasets with globally and locally active variables at the final run. Of the 21 input variables, SOLID identified several as locally active. At the last run, variables Position 1, Acceleration 1, and Acceleration 4 were identified as locally active in 93%, 100%, and 96% of the simulated datasets.

7. Discussion

When optimizing a function $f(\cdot)$ where each evaluation is expensive, one goal is to obtain the largest $f(\hat{\boldsymbol{\chi}})$ in as few evaluations of f as possible. To that end, we proposed SOLID, a new method that measures local variable importance around $\hat{\boldsymbol{\chi}}$ and uses this information to optimize f in a sequential design. Whereas global variable selection permanently removes globally inactive variables, our local variable selection approach is flexible, adapting to the uncertainty of $\hat{\boldsymbol{\chi}}$. We tailored local variable selection to optimize the search for both the maximizer of the AEI acquisition function and the global maximizer. Rather than exploring the entire p-dimensional space, SOLID examines only the locally active variables. In a simulation study, we found that our definition of local importance successfully captured the subset of locally active variables across multiple test functions. By reducing the optimization dimension global and local variable selection, our SOLID algorithm achieved higher $f(\hat{\boldsymbol{\chi}})$ values compared to the standard methods, for a fixed number of sequential evaluations.

One reviewer pointed out that the presence of locally active variables could lead to nonstationary behavior in the response surface. Depending on the nature of nonstationarity, this could result in the estimated GP spatial range parameters, which assume a stationary covariance, as poor indicators of a variable's global importance. It is our intention that any variable that influences f anywhere in the input space be classified as globally

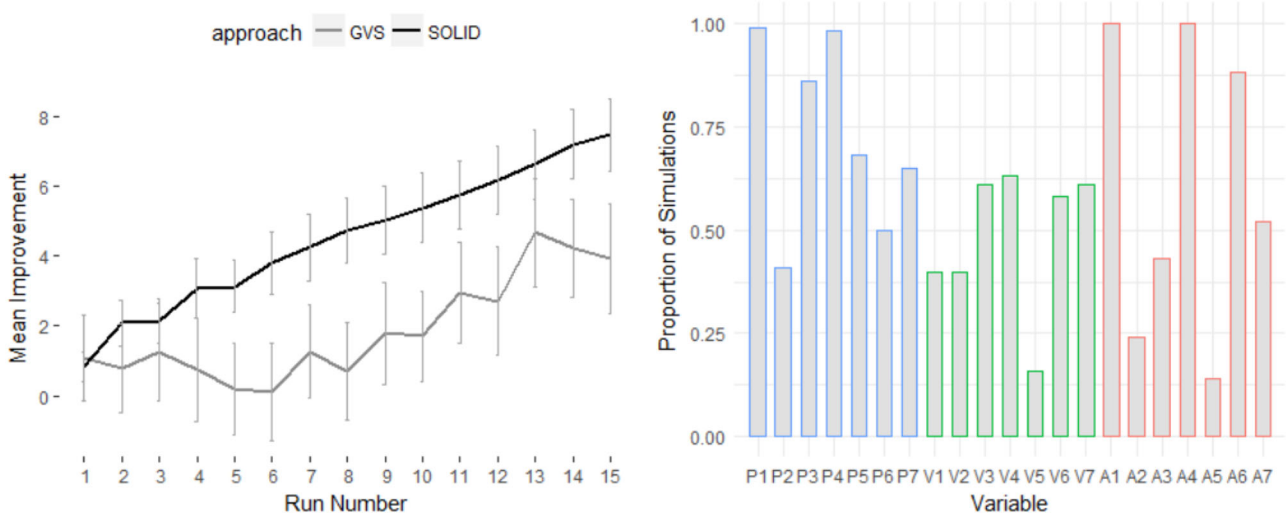

Figure 7. The left plot is mean relative improvement (±1 SE) by run across 100 simulations. The right plot is the proportion of 100 simulated datasets in which each variable was locally active at the 15th run. The variables above correspond to the positions (P), velocities (V), and accelerations (A) of seven locations on the robot arm.

active. Using a GP model with a nonstationary covariance function is the next step, though computational costs would increase and our definition of globally active would need to be modified. On the other hand, a nonstationary covariance function may not be necessary. The three test functions in Section 5 exhibit nonstationary behavior, yet the SOLID behaves well and rarely drops a globally active variable. It is also possible that a stationary covariance function would still able to predict f in a subregion near $\boldsymbol{\chi}$, which is one reason why we chose our local selection criterion to be based on prediction comparisons instead of directly inspecting the estimates of the spatial range parameters.

There are several ways that SOLID could be further improved. In our implementation, we fixed the global and local selection threshold parameters (i.e., g and ρ) but an adaptive approach could lead to improved performance. This is also true for the δ parameter used to declare local importance. Ideally this parameter would shrink as the design space is filled. Finally, there needs to be further exploration of the design impact on SOLID's performance, both in terms of the design size and how well the design supports estimation of the γ_k. Following Linkletter et al. (2006), we used a maximin Latin hypercube design as our initial design. We have started to explore the distance-distributed designs discussed in Zhang, Cole, and Gramacy (2019) as they can better estimate the γ_k.

One limitation of SOLID is its required computations to estimate local importance and its utilization of MCMC to estimate f. In instances where the underlying function is inexpensive to evaluate, it would be faster to use conventional sequential design approaches. Additionally, for experiments involving, say, more than 50 variables, the MCMC algorithm presented here for global and local variable selection could become excessively slow and other methods would be preferable, such as the random embedding approach (Wang et al. 2016) or by specifying an additive model (Kandasamy, Schneider, and Poczos 2015). That said, SOLID's local variable selection could be used for many functions f, without needing to know which or how many variables are locally active. An area of future work would be to incorporate aspects of these other methods within the SOLID framework, perhaps to perform fast initial screening.

Appendix A.

A.1. MCMC Details

We use Metropolis–Hastings within Gibbs sampling to obtain posterior samples of $\boldsymbol{\Omega}$. For convenience, we reparameterize to the total precision (inverse variance), $\eta = (\sigma^2 + \tau^2)^{-1}$, and proportion of variance from the response surface, $r = \sigma^2\eta$. Using the parameterization in Section 2.2, let

$$V(\mathbf{x}, \mathbf{x}') = \frac{1}{\eta}\left[rK(\mathbf{x}, \mathbf{x}') + (1-r)1_{\{\mathbf{x}=\mathbf{x}'\}}\right] \equiv \frac{1}{\eta}W(\mathbf{x}, \mathbf{x}'). \quad \text{(A.1)}$$

Denote $\frac{1}{\eta}\mathbf{W}_\mathbf{X}$ as the n x n covariance matrix corresponding to $\mathbf{X}$. The log-likelihood is

$$\log L\left(\mathbf{y} \mid \boldsymbol{\Omega}, \mathbf{X}\right) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\frac{1}{\eta}\mathbf{W}_\mathbf{X}\right| - \frac{\eta}{2}(\mathbf{y}-\mu\mathbf{1}_n)^T\mathbf{W}_X^{-1}(\mathbf{y}-\mu\mathbf{1}_n). \quad \text{(A.2)}$$

The full conditional distributions of μ, η, θ, and b_k are conjugate, and so these parameters are updated by sampling from their full conditional distributions

$$\begin{aligned}
\eta \mid \text{rest} &\sim \text{Gamma}\left(\frac{n}{2} + a_\eta\ ,\ b_\eta + \frac{1}{2}\left[(\mathbf{y}-\mu\mathbf{1}_n)^T\mathbf{W}_\mathbf{X}{}^{-1}(\mathbf{y}-\mu\mathbf{1}_n)\right]\right)\\
\mu \mid \text{rest} &\sim \text{Normal}\left(\frac{\eta\mathbf{1}_n^T\mathbf{W}_X^{-1}\mathbf{y}}{\sigma_\mu^{-2}+\eta w}\ ,\ \frac{1}{\sigma_\mu^{-2}+\eta w}\right) \qquad \text{(A.3)}\\
\theta \mid \text{rest} &\sim \text{Beta}\left(\alpha_\theta + \sum_{k=1}^p b_k\ ,\ b_\theta + p - \sum_{k=1}^p b_k\right)\\
b_k \mid \text{rest} &\sim \text{Bernoulli}\left(\frac{p_{k1}}{p_{k1}+p_{k0}}\right),
\end{aligned}$$

where $w = \mathbf{1}_n^T\mathbf{W}_X^{-1}\mathbf{1}_n$ and $p_{k\ell} \equiv p(\mathbf{y} \mid b_k = \ell, \boldsymbol{\Omega}_{(-k)})p(b_k = \ell \mid \theta)$ for $\boldsymbol{\Omega}_{(-k)} = \boldsymbol{\Omega}_k/\{b_k\}$.

We implement the Metropolis–Hastings algorithm (Hastings 1970) to update r and $u_1, \ldots, u_p$. The variance ratio r is sampled using the Metropolis–Hastings algorithm with a Beta(10, 1) proposal distribution. For each $k \in \{1, \ldots, p\}$, if $b_k = 0$ then u_k is updated from its prior, otherwise it is updated using a Metropolis–Hastings step with a sliding uniform candidate distribution, conditioned on the current value of u_k, and designed to propose smaller values of u_k corresponding to smoother surfaces. Specifically

$$u_k \sim \text{Uniform}\left(\max\{0, u_k - 50\epsilon(u_k)\}\ ,\ u_k + \epsilon(u_k)\right),$$

where

$$\epsilon(u_k) \equiv \begin{cases} \min\{50, u_k h\} & u_k \geq 30 \\ \max\{1, u_k h\} & 0 \leq u_k < 30 \end{cases} \qquad \text{(A.4)}$$

and $h \sim \text{Unif}(1/2, 2)$. This proposal distribution is used because it ensures candidates are positive and its candidate distribution's variance increases with the current value of u_k.

At each sequential step $i \in \{0, 2, \ldots, N\}$, we estimate $\hat{\boldsymbol{\chi}}^i$ using all p globally active variables and the quasi-Newton optimizer, L-BFGS-B (Byrd et al. 1995), ensuring the maximizer is contained in the design space $[0, 1]^p$. We input the marginal predicted function, $\hat{f}$ and its marginal gradient along with the previous $\hat{\boldsymbol{\chi}}$ (if available) and the four design points with the largest observed responses. This multi-start search is to prevent suboptimal convergence to a local optimal.

After obtaining $\hat{\boldsymbol{\chi}}$, local variable importance is assessed using the posterior draws $\hat{\boldsymbol{\chi}}_t$, and the next design point $\mathbf{x}^*$ is chosen from the $\mathcal{R}^\mathbf{A}$ or $\mathcal{R}^\delta$ restricted spaces. We again used line searches to optimize AEI within these two restricted spaces but other optimization algorithms are possible, including the genetic algorithm and derivative optimizer `genoud` (Mebane and Sekhon 2011). In Step 9 of Algorithm 2, we restrict the search space of $\hat{\boldsymbol{\chi}}$ to lie in $\mathcal{R}^\mathbf{A}$, where each locally inactive variable $j \in \mathbf{A}^c$ is fixed at the corresponding entries of $\hat{\boldsymbol{\chi}}$. This leads to a refined estimate of $\hat{\boldsymbol{\chi}}$, at the end of the each sequential run.

A.2. SOLID Local Tuning Parameters

Two tuning parameters influence the decision to declare a variable locally active or inactive. The δ parameter in (8) affects the spread of the prediction points about $\hat{\boldsymbol{\chi}}_t$, and so the choice of δ affects L_k. Smaller values of ρ allow for a larger number of variables to be declared locally active. It is possible to choose ρ after each run such that a fixed proportion of the globally active variables are considered locally active. One could reason that in "some region of the design space, only a small number of factors are actively influencing the response" (Myers, Anderson-Cook, and Montgomery 2016).

To examine how sensitive SOLID's performance was to different specifications of the ρ local variable selection threshold and the δ

Table A.1. A sensitivity analysis on 100 simulated datasets, showing mean improvement after 7 runs, for 6 combinations of ρ and δ.

	Mean improvement (SE)	
Settings	$\delta = 0.2$	$\delta = 0.6$
$\rho = 0.01$	6.55 (1.21)	9.17 (1.50)
$\rho = 0.05$	6.32 (1.32)	8.63 (1.42)
$\rho = 0.15$	3.60 (1.38)	5.56 (1.05)

NOTE: All initial designs use 21 dimensions and $10 \times 21 = 210$ design points.

radius for local importance, we considered 100 simulations using a two-factor crossed design. Because so few variables were declared to be globally inactive, we disabled the global variable selection feature in the sensitivity analysis. We set the initial design to have size $n = 210$ and considered the improvement after 7 runs, limiting the number of runs due to computational costs. Results in Table A.1 show that a larger radius ($\delta = 0.60$) and conservative threshold ($\rho = 0.01$ or $\rho = 0.05$) provided for the best performance. As long as a conservative local variable selection threshold was chosen, the results were not too sensitive to ρ.

Supplementary Materials

The supplementary materials includes additional details about SOLID, simulation results for more test functions, and three R scripts that: (1) generate test functions, (2) create all necessary functions to perform SOLID, and (3) perform a single iteration of the simulation study.

Acknowledgments

The authors are grateful for the constructive comments of the editors and reviewers that improved this article.

Funding

This work was supported by National Science Foundation grants DGE-1633587 and DMR-1535082.

References

Bai, E., Li, K., Zhao, W., and Xu, W. (2014), "Kernel Based Approaches to Local Nonlinear Non-Parametric Variable Selection," *Automatica*, 50, 100–113. [2,4]

Berk, R. A., Bickel, P., Campbell, K., Fovell, R., Keller-McNulty, S., Kelly, E., Linn, R., Park, B., Perelson, A., Rouphail, N., Sacks, J., and Schoenberg, F. (2002), "Workshop on Statistical Approaches for the Evaluation of Complex Computer Models," *Statistical Science*, 17, 173–192. [1]

Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. (1999), "A Rigorous Framework for Optimization of Expensive Functions by Surrogates," *Structural Optimization*, 17, 1–13. [1]

Brochu, E., Cora, V. M., and de Freitas, N. (2010), "A Tutorial on Bayesian Optimization of Expensive Cost Functions, With Application to Active User Modeling and Hierarchical Reinforcement Learning," arXiv no. 1012.2599. [3]

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995), "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific Computing*, 16, 1190–1208. [11]

Chang, P., Williams, B., Bhalla, K., Belknap, T., Santner, T., Notz, W., and Bartel, D. (2001), "Design and Analysis of Robust Total Joint Replacements: Finite Element Model Experiments With Environmental Variables," *Journal of Biomechanical Engineering*, 123, 239–246. [1]

Crombecq, K., Laermans, E., and Dhaene, T. (2011), "Efficient Space-Filling and Non-Collapsing Sequential Design Strategies for Simulation-Based Modeling," *European Journal of Operational Research*, 214, 683–696. [2]

Djolonga, J., Krause, A., and Cevher, V. (2013), "High-Dimensional Gaussian Process Bandits," *Advances in Neural Information Processing Systems*, 26, 1025–1033. [1]

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004), *Bayesian Data Analysis* (2nd ed.), Boca Raton, FL: Chapman and Hall/CRC. [2]

Gramacy, R. B., and Lee, H. K. H. (2012), "Cases for the Nugget in Modeling Computer Experiments," *Statistics and Computing*, 22, 713–722. [2]

Hahn, D., Allers, R., and Minkoff, R. (1994), *The Lion King*, Burbank, CA: Walt Disney Pictures. [7]

Hastings, W. K. (1970), "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, 57, 97–109. [11]

Huang, D., Allen, T. T., Notz, W. I., and Zeng, N. (2006), "Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models," *Journal of Global Optimization*, 34, 441–466. [3]

Jala, M., Levy-Leduc, C., Moulines, É., Conil, E., and Wiart, J. (2016), "Sequential Design of Computer Experiments for the Assessment of Fetus Exposure to Electromagnetic Fields," *Technometrics*, 58, 30–42. [1]

Jones, D. R., Schonlau, M., and Welch, W. J. (1998), "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, 13, 455–492. [3]

Joseph, V. R., and Hung, Y. (2008), "Orthogonal-Maximin Latin Hypercube Designs," *Statistica Sinica*, 18, 171–186. [2]

Kandasamy, K., Schneider, J., and Poczos, B. (2015), "High Dimensional Bayesian Optimisation and Bandits via Additive Models," arXiv no. 1503.01673. [1,11]

Kleijnen, J. P. C. (2015), *Design and Analysis of Simulation Experiments* (2nd ed.), Cham: Springer Publishing Company, Inc. [3]

Kleijnen, J. P. C., Sanchez, S. M., Lucas, T. W., and Cioppa, T. M. (2005), "State-of-the-Art Review: A User's Guide to the Brave New World of Designing Simulation Experiments," *INFORMS Journal on Computing*, 17, 263–289. [2]

Linkletter, C., Bingham, D., Hengartner, N. W., Higdon, D., and Ye, K. Q. (2006), "Variable Selection for Gaussian Process Models in Computer Experiments," *Technometrics*, 48, 478–490. [2,3,11]

Lizotte, D. J., Greiner, R., and Schuurmans, D. (2012), "An Experimental Methodology for Response Surface Optimization Methods," *Journal of Global Optimization*, 53, 699–736. [1]

Loeppky, J. L., Sacks, J., and Welch, W. J. (2009), "Choosing the Sample Size of a Computer Experiment: A Practical Guide," *Technometrics*, 51, 366–376. [10]

Mebane, W., Jr., and Sekhon, J. (2011), "Genetic Optimization Using Derivatives: The rgenoud Package for R," *Journal of Statistical Software* , 42, 1–26. [3,11]

Močkus, J. (1975), "On Bayesian Methods for Seeking the Extremum," in *Optimization Techniques IFIP Technical Conference Novosibirsk July 1–7, 1974*, ed. Marchuk, G. I., Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 400–404. [3]

Myers, R. H., Anderson-Cook, C. M., and Montgomery, D. C. (2016), *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Wiley Series in Probability and Statistics (4th ed.), New York: Wiley. [11]

Oakley, J. E., and O'Hagan, A. (2004), "Probabilistic Sensitivity Analysis of Complex Models: A Bayesian Approach," *Journal of the Royal Statistical Society*, Series B, 66, 751–769. [2]

Parker, R. (2015), "Efficient Computational Methods for Large Spatial Data," Dissertation, North Carolina State University. [10]

Picheny, V., and Ginsbourger, D. (2014), "Noisy Kriging-Based Optimization Methods: A Unified Implementation Within the DiceOptim Package," *Computational Statistics and Data Analysis*, 71, 1035–1053. [3,5]

Regis, R. G. (2016), "Trust Regions in Kriging-Based Optimization With Expected Improvement," *Engineering Optimization*, 48, 1037–1059. [1,5]

Reich, B. J., Storlie, C. B., and Bondell, H. D. (2009), "Variable Selection in Bayesian Smoothing Spline ANOVA Models: Application to Deterministic Computer Codes," *Technometrics*, 51, 110–120. [2]

Sacks, J., Schiller, S. B., and Welch, W. J. (1989), "Designs for Computer Experiments," *Technometrics*, 31, 41–47. [2]

Santner, T. J., Williams, B. J., and Notz, W. I. (2003), *The Design and Analysis of Computer Experiments*, New York: Springer. [1]

Shan, S., and Wang, G. G. (2010), "Survey of Modeling and Optimization Strategies to Solve High-Dimensional Design Problems With Computationally-Expensive Black-Box Functions," *Structural and Multidisciplinary Optimization*, 41, 219–241. [2]

Vijayakumar, S., D'Souza, A., and Schaal, S. (2005), "Incremental Online Learning in High Dimensions," *Neural Computation*, 17, 2602–2634. [10]

Vijayakumar, S., and Schaal, S. (2000), "Locally Weighted Projection Regression: An $O(n)$ Algorithm for Incremental Real Time Learning in High Dimensional Space," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 1079–1086. [2,10]

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and de Freitas, N. (2016), "Bayesian Optimization in High Dimensions via Random Embeddings," *Journal of Artificial Intelligence Research*, 55, 361–387. [1,11]

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992), "Screening, Predicting, and Computer Experiments," *Technometrics*, 34, 15–25. [2]

Zhang, B., Cole, D. A., and Gramacy, R. B. (2019), "Distance-Distributed Design for Gaussian Process Surrogates," *Technometrics*, 1–13. [11]

Zhao, W., Chen, H.-F., Bai, E.-W., and Li, K. (2018), "Local Variable Selection of Nonlinear Nonparametric Systems by First Order Expansion," *Systems & Control Letters*, 111, 1–8. [2]